تعلم الجافا

خطوة بخطوة

للمهندس : سعد الضبي

تفضلو بزيارة قناتنا على الرابط التالي

https://www.youtube.com/channel/UCinR4wnJdU-zr3yn-9ywZyQ?disable_polymer=true

الدرس الأول

• بنية برنامج JAVA

تنتهج لغة الجافا من حيث البنية العامة لبناء البرنامج طريقة **opp** أو مايسمى ب الكائنات الموجهة (اوبجكت اورينتد بروجرام) عند بنائك لبرنامج جافا يجب عليك اولا ان تقوم بإنشاء **A class**
بالطريقة التالية انظر الشكل المقابل **(1.1)**

Public Class class_Name

(1.1)

والمقصود ب **classNamee** هو اسم الكلاس ويحبذ أن يكون اسم الكلاس يدل على عمل البرنامج والخيار مفتوح لك عند تسمية الكلاس

فمثلا إذا أردنا انشاء برنامج بسيط يقوم بجمع عددين هنا يمكننا أن نقوم بإعطاء اسم للكلاس يدل على عمل البرنامج و هو **Add** عندها سيكون سطر تسمية الكلاس كالآتي انظر الشكل المقابل **(1.2)**

public class add

(1.2)

بعد تسمية الكلاس نقوم بفتح قوس ولا نغلق القوس إلا في نهاية البرنامج وهذا يعني أن الكود الذي سنكتبه في الجافا يكون داخل الكلاس

السطر التالي بعد تسمية الكلاس هو بناء الدالة الرئيسية

والمقصود بالدالة الرئيسية أن هذه الدالة هي نقطة بداية تنفيذ الأوامر
اي ان الأوامر سوف تنفذ بشكل متسلسل من اول سطر في الدالة الرئيسية
وتأتي بنية أو شكل الدالة الرئيسية في برنامج الجافا انظر الشكل المقابل **(1.3)**

Public static void main (string [] args)

(1.3)

المقصود بكلمة **public** هو عام واي متغير أو دالة تعرف على أنها **public** هذا يعني أننا نستطيع الوصول إليها وإعادة إستخدامها من خارج الكلاس
وكلمة **static** يعني أن الدالة أو المتغير الذي يعرف على أنه من نوع **static** يتيح لنا الوصول إليه مباشرة بدون انشاء **Object** وهذا سنتحدث عنه لاحقاً
كلمة **main** وتعني الرئيسي وهذه الكلمة محجوزة فقط للدالة الرئيسية
مابين القوسين هي مصفوفة اسمها **args** من نوع **string** وقد تم بناءها لاستقبال قيم من خارج الكلاس
الشكل التالي في الصورة يوضح كيفية بناء برمج بسيط يقوم بطباعة جملة **hello World** بواسطة أمر الطباعة
لمشاهدة هذا الدرس على اليوتيوب اتبع الرابط التالي

F h)
System.out.println("Hello World");

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

الشكل (1.4)

الدرس الثاني

• المتغيرات في JAVA

تعريف المتغير: نستطيع تعريف المتغير بشكل مبسط على أنه وسيط مؤقت يستخدم لتخزين قيمة معينة ويحجز مساحة معينة في الذاكرة
كمثال عندما يكون لدينا قيمة معينة في البرنامج يجب علينا تخزينها في متغير حتى نستطيع التعامل معها
لنفترض انك تريد ناتج قسمة 1000 /22 كيف ستحصل على الناتج في هذه الحالة يجب عليك تعريف متغير حتى تستطيع إنجاز المهمة والحصول على الناتج لاحظ الشكل التالي
x=1000/2
قمنا بتخزين ناتج قسمة 2/1000 في متغير اسمه x وهذه أمور رياضية بديهية
#انواع_المتغيرات_في_جافا
•الأنواع العددية وهي :
النوع : byte يقبل قيمة عددية ويحجز مساحة 1 بايت في الذاكرة
النوع : short يقبل قيمة عددية ويحجز مساحة 2 بايت في الذاكرة
النوع : int يقبل قيمة عددية صحيحة ويحجز مساحة 4 بايت في الذاكرة
النوع : long يقبل قيمة عددية صحيحة ويحجز مساحة 8 بايت في الذاكرة
النوع : float يقبل قيمة عددية عشرية ويحجز مساحة 4 بايت في الذاكرة
النوع : double يقبل قيمة عددية عشرية ويحجز مساحة 8 بايت في الذاكرة
النوع : char يقبل قيمة حرفية ويحجز مساحة 2 بايت في الذاكرة
النوع : boolean نوع منطقي وهو يقبل إحدى قيمتين فقط اما true أوfalse
#طريقة_تعريف_المتغير
لتعريف المتغير في جافا يجب عليك اولا ان تكتب نوع المتغير ثم اسم المتغير
مثال : لتعريف متغير من نوع عدد صحيح سيكون تعريفه بالشكل التالي

```
int x;
```

بعد ذلك يمكنك إسناد قيمة للمتغير بالشكل التالي

```
x=5;
```

#ملاحظة الفاصلة المنقوطة ؛ تستخدم في نهاية كتابة الأمر في جافا
يمكنك في لغة جافا إسناد القيمة مباشرة عند تعريف المتغير بالشكل التالي

```
int x=5;
```

لتعريف متغير عشري في جافا يكون بالشكل التالي

```
double d=5.60;
```

*** لتعريف متغير حرفي يكون بالشكل التالي**

```
char s='N';
```

*** لتعريف متغير من نوع سلسلة حرفية في جافا نستخدم النوع String بالشكل التالي**

```
String s2 = " اهلا بكم في صفحة المبرمج العربي";
```

لتعريف متغير منطقي يكون بالشكل التالي

```
boolean b=true;
```

ملاحظة ليس من المهم إسناد قيمة للمتغير عند تعريفه يمكنك إسناد قيمة له وقتما تشاء في اي مكان في البرنامج في البرنامج التالي تم تعريف ثلاثة متغيرات من نوع عدد intt ثم تم إسناد قيمة لكل متغير بعد ذلك قام بجمعهم وطباعة ناتج الجمع بإستخدام الأمر

System.out.println();

```
public static void main(String[] args) {

    int first_number, second_number, answer;

    first_number = 10;
    second_number = 20;
    answer = first_number + second_number;

    System.out.println("Addition Total = " + answer );
}
```

الشكل (1.5)

الدرس الثالث

• أوامر الإدخال والإخراج في JAVA

تحدثنا في الدرس السابق في جافا عن المتغيرات
وفي درسنا هذا سوف نتحدث عن أوامر الإدخال والإخراج في لغة جافا
•أوامر الإخراج
-لإظهار النتيجة على الشاشة في لغة جافا
نستخدم الأمر في الشكل المقابل (1.6)

```
System.out.println("yes");
```

الشكل (1.6)

وفي أول درس تعلمنا كيفية طباعة جملة بسيطة Hello World بواسطة الأمر

```
System.out.println("Hello World");
```

سنتحدث الآن عن كيفية إدخال البيانات إلى جافا بواسطة الأمر Scanner
وحتى نستطيع استخدام الأمر Scanner يجب علينا تضمين الحزمة التالية

```
java.util.Scanner;
```

في بداية البرنامج بالشكل التالي

```
import java.util.Scannerr;
```

عد تضمين الحزمة في بداية البرنامج نستطيع إستخدام الأمر Scanner لإدخال القيم ولإستخدام الأمر Scanner يجب عليك أن تقوم بإشتقاق كائن من Scanner كما في الطريقة المقابلة

```
Scanner obname=new Scanner(System.in);
```

المقصود بـ obname هو الكائن الذي سنقوم بإشتقاقه من الكلاس Scanner
حتى نستطيع بواسطة هذا الكائن إستخدام أوامر الإدخال الموجودة داخل الكلاس Scanner تستطيع ان تسمي الكائن هذا بأي إسم تريده
أما البارامتر System.in فهو لإعطاء أمر للجافا بأننا نريد أن نقوم بعملية إدخال
على سبيل المثال إذا أردنا أن نقوم بإدخال الرقم 5 لتطبيق ذلك اتبع الخطوات التالية -11- تعريف كائن من الكلاس Scannerسنعطي هذا الكائن أي إسم على سبيل المثالsaad

```
Scanner saad=new new Scanner(System.in);
```

2-قمنا بتعريف كائن اسمه saad من كلاس أمر الإدحال Scanner وبواسطة الكائن saad سيكون بإستطاعتنا الوصول إلى كافة أوامر الإدخال المتنوعة والمضمنة داخل الكلاس Scanner
3-بما أننا نريد أن ندخل امر صحيح يجب علينا أن نقوم بتعريف متغير من نوع int وتخزين القيمة من الكائن saad المشتق من الكلاس Scanner بالطريقة التالية

```
Int n=saad. .nextInt();
```

#لاحظ إستخدمنا دالة الإدخال ;()nextInt

لأننا نريد أن نقوم بإخال صحيح
على سبيل المثال لنفترض أننا نريد إدخال سلسلة نصية
في هذه الحالة سوف نقوم بتعريف متغير من نوع
بالطريقة التالية

String s= saad.nextLine();

#لحظ إستخدمنا **nextLine();** لأننا نريد أن ندخل سلسلة نصية
على سبيل المثال أيضاً لنفترض أننا نريد أن ندخل قيمة من نوع **byte** سيكون أمر الإدخال بالطريقة التالية

byte id= saad.nextByte();

وهكذا
عمل البرنامج الذي في الصورة انه يطلب من المستخدم ادخال الاسم والعمر ورمز **Id** ثم يقوم بطباعة الناتج على الشاشة
ويمكنك ملاحظة كيف تعامل مع دالة الإدخال

المثال التالي يوضح كيفية إدخال البيانات وطباعتها

```
package javaapplication2;
import java.util.Scanner;//تضمين حزمة الإدخال
class TesInput
{
 public static void main(String []args)
 {
    String Name="";
    int age=0;
    byte id=0;
    Scanner n=new Scanner(System.in); // إنشاء كائن للإسم
    Name=n.nextLine();
    Scanner n1=new Scanner(System.in);// إنشاء كائن للعمر
    age=n1.nextInt();
     Scanner n2=new Scanner(System.in);// إنشاء كائن للآيدي
    id=n2.nextByte();
    System.out.println("Name="+Name);
    System.out.println("age="+age);
    System.out.println("id="+id);
 }}
```

الشكل (1.6)

الدرس الرابع

الشرطية#IF_Else

تستخدم IF الشرطية لفحص قيمة أو عدة قيم وبناءً على نتيجة الفحص تستطيع إتخاذ القرار المناسب وتوجيه مسار البرنامج

#تركيبة if الشرطية في جافا كما في الشكل التالي

```
If(هنا يكتب الشرط)
```

على سبيل المثال إذا أردنا كتابة برنامج يطلب من المستخدم إدخال عدد زوجي فإذا أدخل المستخدم قيمة غير زوجية تظهر له رسالة تفيد بأن القيمة المدخلة غير مقبولة

```
If(num%2!=0)
sysytem.oout.println("القيمة المدخلة غير مقبولة");
```

#if_else

يتم استخدام if_elseee في حالة توقع إحدى نتيجتين حسب الشرط المعطى فإذا تحقق الشرط فإنه يتجاهل else وعند عدم تحقق الشرط فإنه يتجاهل if ويتم تنفيذ else

على سبيل المثال إذا أردنا فحص نتيجة الطالب إذا كانت أكبر من أوتساوي الخمسين يطبع ناجح مالم إذا كانت درجة الطالب أقل من 50 يطبع راسب

```
If(deg>=50)
sysytem.oout.println("ناجح");
Else
sysytem.oout.println("راسب");
```

If -الشرطية المتداخلة

نستخدم if الشرطية المتداخلة في حالة إذا كان لدينا عدد لا محدود من الشروط وتأتي تركيبة if الشرطية بالشكل التالي

```
If(الشرط)
// code
Else if(شرط)
//code
```

فعلى سبيل المثال إذا أردنا فحص درجة حرارة الجو بالشكل التالي

-إذا كانت درجة حرارة الجو تساوي صفر اطبع بارد جداً

-إذا كانت درجة حرارة الجو بين صفر و 10 اطبع بارد

```
If(val==0)
sysytem.oout.println("بارد جداً");
Else if(val>0 && val<10)
sysytem.oout.println("بارد");
```

الصورة المرفقة مع البرنامج تقوم بطلب من المستخدم إدخال قيمة ثم يتم فحص القيمة فإذا كان باقي قسمة العدد / 2 = صفر فإن النتيجة تطبع زوجي

مالم إذا لم يتحقق الشرط فإن النتيجة تكون فردي

```
package javaapplication2;
import java.util.Scanner; // إسم حزمة أوامر الإدخال
public class JavaApplication2 {
  public static void main(String[] args) {
     Scanner input = new Scanner(System.in);
    int num;
    System.out.println("Enter the Number");
    num=input.nextInt();
    if(num%2==0)
      System.out.println(num + "  " + " is even");
    else
       System.out.println(num + "  " + " is odd");
  }}
```

الشكل (1.7)

الدرس الخامس

#الأمر_ switch case دالة الإحتمالات

يتم إستخدام الأمر **switch case** **هو شبيه بالأمر الشرطي IF الذي تكلمنا عنه في الدرس السابق**
وتستخدم دالة switch caseee عندما يكون لدينا عدة إحتمالات للقيمة المدخلة من المستخدم وبناءً على كل احتمال نقوم بتنفيذ مهمة معينة
على سبيل المثال
لنفترض ان المستخدم سوف يدخل إحدى 3 قيم
إذا أدخل الرقم 1 يظهر له رصيده البنكي
وإذا أدخل الرقم 2 يظهر له المسحوبات خلال السنة
وإذا أدخل الرقم 3 يتم إنهاء العملية وإنهاء البرنامج
من المثال السابق يتضح لنا أننا نحتمل من المستخدم إدخال إحدى 33 قيم وبناءً على القيمة المدخلة نقوم بتنفيذ حدث معين

وتأتي تركيبة ال switch case بالشكل التالي

```
switch(expression) {
case constant-expression :
statement(s);
break;
case constant-expression :
statement(s);
break; /* optional */
/* you can have any number of case statements */
default : /* Optional */
statement(s);
```

تستخدم **switch** لإستقبال القيمة المدخلة ثم فحصها بواسطة الأمر **casee** وبناءً على النتيجة التي تظهر من **case** يتم اتخاذ القرار على سبيل المثال

```
String x;
X= input.nextInt() ;
Switch(x)
case "11":
System.out.println "Hello");
break;
case "22":
System.out.println ("good night");
break;
default:
System.out.println
("Invalid gradee");
```

لاحظ تام وضع قيمة ال **X** داخل ال **Switch** ثم فحصها بواسطة **casee** الأحتمال الأول إذا أدخل المستخدم الرقم **1** يتم طباعة **Hello** ثم الخروج من البرنامج بواسطة الأمر **break** التي تعني أنه تحقق الإحتمال ولن يتم فحص باقي الإحتمالات أما كلمة **default** فيتم تنفيذها في حالة واحدة فقط وهو أن المستخدم أدخل قيمة غير متوقعة نقوم بإظهار رسالة له تفيد بأن القيمة المدخلة خاطئة أو غير متوقعة

في البرنامج التالي الموجود في الصورة

يطلب من المستخدم إدخال رقمين تدخل الرقم ثم تضغط **Enterrr** ثم تدخل الرقم الثاني ثم تضغط **Enter** بعد ذلك يطلب منك إدخال أحدى العمليتين فإذا أدخلت علامة **+** سوف يتم جمع العددين المدخلة وإذا أدخلت علامة **–** سوف يتم طرح العددين المدخلة وهكذا

في حالة إدخالك ل علامة غير موجودة ضمن **caseee** وهي قسمة مثلاً لن يتم تنفيذها لأنها غير مكتوبة ضمن ال **case** في هذه الحالة سيقوم بإعطائك رسالة تفيد بأن المدخل خاطيء (ينبغي عليك بعد قراءة الدرس ومشاهدة المثال الذي في الصورة أن تقوم بإضافة عمليات مثل القسمة والضرب والأسس(

```
package javaapplication2;
import java.util.Scanner; // إسم حزمة أوامر الادخال
public class JavaApplication2 {
  public static void main(String[] args)
  {
    Scanner input = new Scanner(System.in);
    String op;
   int num1,num2,result;
   System.out.println("Please Enter 2Number ");
    num1 = input.nextInt();
    num2 = input.nextInt();
    System.out.println("Please Enter Operand(-,+)");
    op=input.next();
    switch(op)
    {
      case "+":
        result=num1+num2;
        System.out.println(result);
        break;
        case "-":
        result=num1+num2;
        System.out.println(result);
        break;
    }
```

الدرس السادس

#حلقات التكرار Loops

في الدرس السابق تحدثنا عن الأوامر الشرطية وتعرفنا على جملة Switch
في هذا الدرس أحبتي في الله سنتعرف على حلقات التكرار في البرمجة
تستخدم حلقات التكرار لتكرار أمر معين
على سبيل المثال نريد طباعة بيانات خمسين طالب هل سنكتب امر الطباعة خمسين مرة !
يبدو الأمر مجهداً ومملاً وربما نترك البرمجة لهذا السبب
بدلاً من ذلك سنكتب سطراً واحداً فقط بإستخدام إحدى أوامر حلقات التكرار وبذلك نوفر على أنفسنا الوقت والجهد

مثال أخر نريد أن نقوم بإيجاد ناتج المضروب لعدد معين هل نقوم بكتابة تسلسل الرقم كل مرة
تخيل لو أردنا إيجاد ناتج المضروب للعدد خمسين هل سنقوم بكتابة عشرات الأسطر لإيجاد الناتج !!
لهذا يتضح لنا أهمية حلقات التكرار التي ستجد انك لن تستغني عنها مع كل برنامج تقوم بإنشاءه
وتعرف حلقات التكرار بأنها دوال تقوم بعمل حلقات تكرارية
كما في الشكل المقابل في لغة الجافا لدينا ثلاث أنواع لحلقات التكرار وهي

1- for
2- while
3- do....while

وكل هذه الأنواع تقوم بنفس العمل مع اختلاف في بعض الخصائص يمكنك إستخدام أياً منها في عملك

- #بنية حلقة for

كما تلاحظ في البنية for بأنه يتم كتابة الكلمة المحجوزة forr ثم قوس مفتوح وبين القوسين يتم كتابة شرط دوران الحلقة
•التعبير initialization يعني بداية الحلقة أو بداية الدوران
•التعبير Boolean_expression يعني نهاية الحلقة
•التعبير update ويعني مقدار الزيادة

```
for(initialization;
Boolean_expression; update) {
// Statements
}
```

على سبيل المثال إذا أردنا أن نطبع الكلمة ("Hello ")خمس مرات سيكون إستخدام ال forكما في الشكل المقابل

```
For(int i=0;i<5;i++)
System.out.println("Hello");
```

كما تلاحظ في الشكل السابق بأانا في التعبير initializationقمنا بتعريف متغير اسمه i من نوع intوبدأناه من الصفر

وفي قسم التعبير Boolean_expressionقمنا بتحديد عدد حلقات الدوران وهو أن لا يزيد عى خمس لأننا نريد أن نطبع الكلمة hello خمس مرات

بنية while

كما تلاحظ في الشكل السابق تم استخدام الكلمة المحجوزة while ثم قوس مفتوح وقوس مغلق بين القوسين نقوم بكتابة الشرط الذي سينتهي به حلقات التكرار ويسمى بالشرط المنطقي كما في for
بعد كتابة الشرط نقوم بكتالة التعبير داخل بنية ال {} while
على سبيل المثال إذا أردنا طباعة جملة (أهلاً بكم مجموعة عائلة البرمجة) خمس مرات
سيكون بناء البرنامج بالشكل التالي

```
while (Boolean expression)
{
statement(s) //block of statements
}

Int i=0;
While(i<5)
{
System.out.println(" أهلاً بكم مجموعة عائلة البرمجة ");
Il++;
}
```

#بنية do while

في أمر التكرار do while يتم استخدام الكلمة المحجوزة do ثم يتم بناء ال blockk ثم كتابة الكلمة المحجوزة while
الفرق بين ال while و do while أن ال whileتفحص الشرط ثم تنفذ الأمر حسب حلقات التكرار
أما do while فهي تنفذ ثم تفحص الشرط
على سبيل المثال إذا أردنا طباعة تسلسل العدد إبتدأ من 10 حتى 19
سيتم كتابة البرنامج بالشكل المقابل

```
do {
// Statements
}while(Boolean_expression);

do {
System.out.print("value of x : " + x );
x++;
System.out.print("\n");
}while( x < 20 );
```

#الدرس_السابع

#المصفوفات_أحادية_البعد

#تعريف المصفوفة : المصفوفة هي عبارة عن تقنية أو أداة لخزن البيانات من نوع واحد بشكل مؤقت وهي مقسمة لعدة خلايا وكل خلية لها عنوان index لذلك يسهل علينا معالجة البيانات وترتيبها وفرزها والبحث فيها باستخدام المصفوفات

والمصفوفة لا تقبل إلا نوع واحد من البيانات يعني أن تكون كل البيانات المخزنة في المصفوفة من نوع واحد إما عددي فقط او نصية فقط وهكذا.......

الشكل التالي يوضح كيفية تعريف المصفوفة ذات بعد

arrayType []ArrayName=new ArrayType[siz];

- المقصود بـ arrayTypee وهو نوع المصفوفة ويتم تحديد نوع المصفوفة حسب نوع البيانات التي سيتم تخزينها في المصفوفة وكما قلنا سابقاً بأن المصفوفة لا تقبل سوى نوع واحد من البيانات أي لا يمكننا إدخال بيانات من نوع stringوبيانات من نوع int في مصفوفة
- المقصود بـ ArrayName وهو اسم المصفوفة
- الكائن new لتحرير مساحة في الذاكرة للمصفوفة بنفس حجم المصفوفة

على سبيل المثال إذا أردنا أن نقوم بإنشاء مصفوفة من نوع int اسمها a ستكون بالشكل المقابل

Int a=new int [5];

في المثال السابق قمنا بإنشاء مصفوفة وحجمها 5

مثال أخر نريد إنشاء مصفوفة من نوع doubleاسمها d وحجمها 100 سيكون تعريفها بالشكل المقابل

Double d=new double[10];

المثال في الصورة المرفقة مع الشرح يوضح كيفية تعريف مصفوفة أحادية البعد من نوع intt ثم إدخال عناصر وتخزينها في المصفوفة وطباعتها

```
package javaapplication5;
// عائلة البرمجة
import java.util.Scanner;// تضمين حزمة الإدخال
public class JavaApplication5 {

    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);// تعريف كائن الإدخال
        int []a=new int[5]; // تعريف المصفوفة وتحديد الحجم
        for(int i=0;i<a.length;i++)
            a[i]=input.nextInt(); // طريقة الإدخال
        for(int j=0;j<a.length;j++)
            System.out.println(a[j]); //   طباعة محتويات المصفوفة
    }}
```

#الدرس_الثامن

#المصفوفة_ذات_بعدين Array 2D

تحدثنا في الدرس السابق عن المصفوفة ذات بعد والتي تتكون من عمود واحد مقسم إلى عدة خلايا وكل خلية لها عنوان بحيث نستطيع بواسطة هذه العناوين التعامل مع خلايا المصفوفة

سنتحدث اليوم عن المصفوفة ذات بعدين تتكون المصفوفة ذات بعدين من مجموعة من الأعمدة والصفوف المتقاطعة مشكلة بتقاطعها مايسمى بالخلايا وأشبه ماتكون بالجدول ويتم تعريف مصفوفة ذات بعدين في لغة جافا بالشكل المقابل

type [] [] Array_Name =new Type[][]

المقصود بـــ type هو نوع المصفوفة
المقصود بـ Array_Name هو اسم المصفوفة

على سبيل المثال إذا أردنا تعريف مصوفة ذات قيم عددية من نوع intt سيكون تعريفها بالشكل المقابل

int [][]a=new int [size][sizee]

من أهم الأشياء التي لا بد أن نعرفها في المصفوفات وخاصة المصفوفة ذات بعدين هو أن نعرف أن لكل خلية عنوان فإذا نظرنا إلى عنوان الخلية في المصفوفة ذات بعدين سنجده يتألف بالشكل التالي
(I,j)بحيث يشير ال i إلى الأعمدة وال jj يشير إلى الصفوف فعلى سبيل المثال إذا كان لدينا خلية عنوانها (1,5) فهذا يشير إلى أن ال i=1 و j=5
فعلى سبيل المثال إذا أردنا تخزين القيمة 1000 في الخلية (1,3) في مصفوفة ذات بعدين اسمها a نستطيع تطبيق ذلك بالشكل المقابل

A(1,3)=1000

بالنظر إلى الشكل المقابل نستطيع فهم بنية المصفوفة ذات بعدين بشكل أكثر وضوحاً

0.1	0.2	0.3	0.4
1.1	1.2	1.3	1.4
2.1	2.3	2.4	2.4
3.1	3.2	3.3	3.4

فمن الشكل السابق نستطيع ملاحظة عنوان كل خلية فعلى سبيل المثال إذا نظرنا إلى الصف الأول
|(0,0)|(0,1)| (0,2)|(0,3)|
سنلاحظ أن قيمة ال I ثابتة بينما ال j هي التي تتغير لأنه كما قلنا سابقاً بأن ال ii تشير إلى الأعمدة وال j تشير إلى الصفوف
وإذا نظرنا إلى العمود الأول
|(0,0)|
|(1,0)|
|(2,0)|
|(3,0)|
سنجد أن قيمة ال i تتغير بينما قيمة ال j هي الثابتة لأنه كما قلنا سابقاً بأن ال ii تشير إلى

الأعمدة وال j تشير إلى الصفوف
فعلى سبيل المثال إذا أردنا جمع الصف الثالث
سنجد أن قيمة ال i في الصف الثالث قيمتها 22
ثابتة لا تتغير بينما قيمة ال j هي التي تتغير
يمكننا تنفيذ ذلك بواسطة الأمر التالي
X=a(2,j)
البرنامج التالي في الصورة المرفقة يوضح كيفية
تعريف مصفوفة
*قراءة مصفوفة.
* طباعة مصفوفة.

```
package javaapplication4;
//عائلة البرمجة
import java.util.Scanner;
public class JavaApplication4 {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        int [][]a=new int [4][4]; //   تعريف مصفوفةمع تعيين حجمها
        int i,j;
        for( i=0;i<4;i++)
            for( j=0;j<4;j++)
            a[i][j]=input.nextInt();
        for( i=0;i<4;i++){
            for(j=0;j<4;j++)
          System.out.print(a[i][j]);
         System.out.println();
        }
    }}
```

#الدرس_التاسع

•#**معالجة_الإستثناءات_بواسطة** Try – catch

ماهي الإستثناءات في البرمجة : هي أخطاء غير متوقعة تحدث من قبل المبرمج أو من قبل المستخدم مما يؤدي إلى توقف عمل البرنامج.

•أنواع الإستثناءات الشائعة

-استثناء رياضي مثل القسمة على صفر

ArithmeticException Arithmetic error, such as divide-by-zero.

-إستثناء الطفح ويحدث هذا في المصفوفات

- ArrayIndexOutOfBoundsException

وغالباً مايحدث إستثناء الطفح في المصفوفات على سبيل المثال إذا افترضنا أن لدينا مصفوفة حجمها 6 بالشكل المقابل

```
Int [] a=new int[6];
A[7]=5;
```

في هذه الحالة سوف يحدث إستثناء من نوع IndexOutOfBounds لإن المصفوفة حجمها 6فالعدد 7 خارج عن نطاق المصفوفة في هذه الحالة سوف يحدث بما يسمى الطفح

-أخطاء القيمة null عند مساواة متغير بمتغير آخر ولكن هذا المتغير لا توجد فيه قيمة أي null

مثل هذه الإستثناءات تستطيع معالجتها وتصيدها بإستخدام

try – catchh

تركيبة try – catchh

```
try
{
//statements that may cause an
exception
}
catch (exception(type) e(object))
{
//error handling code
 }
```

في منطقة

في منطقة try نضع الكود الذي نشك أن يحدث فيه إستثناء أما مهمة Catch

فهي طباعة رسالة تظهر نوع الإستثناء والفائدة من ذلك أن try – catch سوف تصطاد الخطأ وتمنعه من إحداث تعليق أو تهنيق في البرنامج

لنشاهد مثالاً بسيطاً على بعض العمليات التي تؤدي لحدوث الإستثناء انظر الشكل المقابل

```
public static void main(String[] args)
{
Scanner s= new Scanner(System.in);
int a=s.nextInt();
int b=s.nextInt();
int c=b/a;
System.out.println(c);
}
```

لنفترض ان المستخدم أدخل قيمة a صفر ي هذه الحالة سوف يحدث إستثناء

مما يؤدي إلى توقف البرنامج عن العمل

نستطيع معالجة الإستثناءات وتصيدها بواسطة try – catch انظر الشكل المقابل

```
try{
int a=s.nextInt();
int b=s.nextInt();
int c=b/a;
System.out.println(c);
}catch (Exception e)
{System.out.println(e.getMessage());
}
```

لاحظ وضعنا الكود الذي نشك بحدوث إستثناء فيه في منطقة try لتصيد الإستثناء والقضاء عليه قبل أن يحدث لنا مشكلة أما catch مهمتها طباعة نوع الخطأ بواسطة الأمر e.getMessage()

2-استخدام finally تستخدم finally مع try-catchلتنفيذ أمر معين سواءً حدث إستثناء أو لم يحدث على سبيل المثال إذا أردنا طباعة جملة في البرنامج السابق أي جملة على سبيل المثال الجملة التالية" Math Program "

```
try{
int a=s.nextInt();
int b=s.nextInt();
int c=b/a;
System.out.println(c);
System.out.println("Math Program");
}catch (Exception e)
{System.out.println(e.getMessage());}
```

#لاحظ سيتم طباعة الجملة " Math Program " في حال عدم حدوث إستثناء ولكن عند حدوث إستثناء سيتم الخروج من البرنامج ولن يتم طباعة الجملة أو تنفيذ بقية الأسطر
ولكن إذا أردنا من البرنامج طباعة هذه الجملة إجبارياً سواءً حدث إستثناء أم لم يحدث في هذه الحالة سوف نستحدم finally وسوف نقوم بالتعديل على الكود بالطريقة التالية

```
try{
int a=s.nextInt();
int b=s.nextInt();
int c=b/a;
System.out.println(c);
}catch (Exception e) {System.out.println(e.getMessage());}
Finally{ System.out.println("Math Program");
}
```

هنا سيتم تنفيذ أمر الطباعة مع ال finally سواءً حدث إستثناء أم لم يحدث

```
package javaapplication7;
import java.util.Scanner;
public class JavaApplication7 {
    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        int a,b,c;
        try{
         a=s.nextInt();
         b=s.nextInt();
         c=b/a;
        System.out.println(c);

        }catch (Exception e) {System.out.println(e.getMessage());}
        finally{System.out.println("math Program");}
        }
    }  // عائلة البرمجة
```

#الدرس_العاشر

#الدوال

#كيفية_بناء_الدوال

#تعريف_الدالة : الدالة هي مجموعة من الأوامر البرمجية تقوم بتنفيذ مهمة معينة.

#لماذا_نستخدم_الدوال؟

يتم استخدام الدوال في البرمجة لتنظيم عمل البرنامج وتقليل عدد الأسطر في البرمجة على سبيل المثال إذا افترضنا ان هناك أمر برمجي نحتاجه أكثر من مرة في برنامجنا هل سنظل نكتب هذه الأمر أكثر من مرة بالطبع لا
سنقوم بكاتبة هذه الأسطر التي نحتاجها داخل دالة ثم نستدعي الدالة وقتما نشاء وبذلك نكون قد قمنا بتقليل عدد الأسطر البرمجية وقمنا بجعل الكود أكثر تنظيماً وأكثر فهماً

#أنواع_الدوال

1- دوال تعيد قيمة
2- دوال لا تعيد قمية

#دوال_تعيد_قيمة

يمكنك أن تقوم ببناء دالة تعيد قيمة وتستدعيها من أي مكان في البرنامج ويأتي شكل الدالة في برنامج الجافا بالشكل المقابل

المقصود بـ Type هو نوع القيمة المرتجعة من الدالة فإذا كانت القيمة عددية صحيحة يكون نوع الدالة int وإذا كانت القيمة عددية عشرية يكون نوع الدالة double وإذا كانت نوع القيمة المرتجعة نص يكون نوع الدالة string وهكذا المقصود Funcatio_Name هو اسم الدالة وهذا الإسم إختياري ضع أي إسم تشاء ولكن يجب أن يكون الإسم يدل على عمل الدالة

```
Type Funcatio_Name(parameter)
{
}
```

```
Int sum(int n1,int n2)
{
return(n1+n2);
}
```

المقصود بـ هي القيمة التي سوف تستقبلها الدالة وهي اختيارية يمكنك أن تقوم ببناء دالة لا تستقبل قيم أو العكس وهذه القيم يتم أرسالها عند إستدعاء الدالة
على سبيل المثال إذا أردنا بناء دالة تستقبل عددين وتعيد ناتج جمعهم
سيكون بناء الدالة بالشكل المقابل

```
Int sum(int n1,int n2)
{
Int x=n1+n2;
return x;
}
```

#لاحظ قمنا ببناء دالة تستقبل عددين n1,n2 ثم تعيد ناتج جمعهم عن طريق الأمر return
والأمر return يعني إعادة القيمة المستخلصة من الدالة ثم إنهاء عمل الدالة
يمكننا إستدعاء الدالة في البرنامج الرئيسي بالشكل المقابل

```
Int r=sum(5,3);
```

تم إستدعاء الدالة sum وإرسال رقمين 5 و 3 هنا سوف تستقبل معاملات الدالة أو parmatersالدالة هذه الأعداد وسوف يتم تخزينها في البارامترات أي أن n1 سوف تستقبل العدد 5 والبارامتر n2 سوف يستقبل العدد 3 ، وسوف يتم إعادة ناتج جمعهم بواسطة الأمر return (n1,n2);
قد يبدو الأمر غامضاً قليلاً مع استخدام الأمر returnسنقوم بإعادة صياغة الدالة بشكل أوضح

#لاحظ قمنا بتعريف متغير محلي إسمه x يحتوي على ناتج جمع العددين التي ستستقبلهما الدالة وبعد ذلك ثمنا بإعادة قيمة المتغير x إلى الدالة sum ثمة أمر مهم لا بد أن تعرفه وهو أننا نعيد القيمة المستخلصة من عمل الدالة دائماً

#دالة لا تعيد قيمة

في كثير من الأحيان نقوم ببناء دالة تقوم بعمل معين ولا ننتظر منها أن تعيد قيمة على سبيل المثال نقوم ببناء دالة تقوم بالتعديل في قيمة معينة فقط أو نقوم ببناء دالة تقوم بتخزين البيانات في الجدول وهكذا
الدوال التي لا تعيد قيمة يكون نوعها void على سبيل المثال ولن نذهب بعيداً سنبقى في الدالة sumإذا أردنا من الدالة sum أن تقوم بجمع العددين المرسلة إليها وطباعة الناتج مباشرة بدون إعادة الناتج بواسطة الأمر return سنعيد صياغة الدالة بالشكل المقابل

```
Void sum(int n1,int n2)
{
Int x=n1+n2;
System.out.println(x);
}
```

في الشكل المقابل للدالة استبدلنا نوع الدوال من Intإلى void لأننا لانريدها أن تعيد قيمة وبدلاً من الأمر return كتبنا دالة الطباعة
في برنامج الجافا يتم بناء الدوال تحت اسم الكلاس class مباشرة وبهذا تكون الدوال التي نبنيها في الجافا ملكاً للكلاس ولا نستطيع إستدعاء الدالة إلا بإذن من الكلاس ولأخذ إذن من الكلاس للوصول إلى الدوال الخاصة به نقوم بإنشاء object كائن من اسم الكلاس وعن

طريق هذا الكائن object نستطيع الولوج أو الوصول إلى الدوال والمتغيرات الخاصة بالكلاس

الصورة المرفقة مع الشرح توضح كيفية بناء دالة ومن ثم استدعائها من البرنامج الرئيسي مع أعطائها قيم أو إرسال قيم إلى الدالة

الخطوة الأولى قمنا بإنشاء كائن object اسمه sمن الكلاس Saad1 بالطريقة التالية

Saad1 s=new Saad1();

الخطوة الثانية وبإستخدام الكائن s قمنا بإستدعاء الدالةsum

```
package saad1;
public class Saad1 {
     int sum(int n1,int n2)
    {
        return(n1+n2);
    }
    public static void main(String[] args) {
        Saad1 s=new Saad1();
        int n=s.sum(5,3);
        System.out.println(n);
    }

}//  عائلة البرمجة
```

الدرس الحادي عشر

البرمجة بالكائنات الموجهة Object Oriented Programm

ماهو ال object oreinted بشكل مبسط وسلس

ال object oriented نستطيع أن بأنها تقنية يستفيد منها المبرمج في الأشياء التالية

- **حماية البيانات** وهو مايسمى بتغليف البيانات حيث يستطيع المبرمج تعريف المتغيرات والدوال داخل كلاس بدلا من كتابتها بالطريقة العادية عندما يقوم المبرمج بتعريف المتغيرات وبناء الدوال داخل الكلاس سيستفيد شيئين اثنين

1-حماية البيانات والتحكم بالطرق الوصول إلى المتغيرات والدوال التي سينيها داخل الكلاس

22-عندما يقوم المبرمج بتعريف المتغيرات والدوال داخل الكلاس فإنه يستطيع إعادة استخدامها بدلا من إعادة كتابتها من جديد وبهذا سيوفر على أنفسنا(الوقت - الجهد - العناء - تكرار الكود) وسنكسب فوق كل هذا أن الكود الذي سنكتبه بطريقة ال object oriented سيكون منظماً ومرتباً

على سبيل المثال لنفترض ان لدينا 200 نموذج في كل هذه النماذج سنحتاج إلى دالة اسمها GetMaxID فهل يعقل اننا سنقوم بكتابة الدالة في كل نموذج ! سنقوم في هذه الحالة بكتابة الدالة مرة واحدة داخل كلاس ثم إعادة استخدامها في كل نموذج وهذا سيوفر علينا الوقت والجهد وسنستفيد أيضاً اختصار الكود

- **الكلاس Class** وهذا ماذكرناه أعلاه ونستطيع تعريف ال Classss بأنه قالب يحتوي على متغيرات Attributes ودوال ال behavior
- **الكائن** وهو نسخة من الكلاس يتم اشتقاقه من نفس الكلاس للوصول إلى المتغيرات والدوال الموجودة داخل الكلاس لإعادة استخدامها

على سبيل المثال لنفترض ان لدينا كلاس اسمه Car

إذا أردنا أن نستخدم المتغيرات والدوال الموجودة داخل الكلاس Carrr يجب علينا أن نقوم بإنشاء كائن من هذا الكلاس كما في السطر التالي

```
Car c1;
```

حيث أن c1 هو كائن مشتق من الكلاس Car وبهذا يمتلك الكائن c111 الحق في الوصول إلى المتغيرات والدوال الموجودة داخل الكلاس Car لنفترض أيضاً أننا نريد استخدام دالة اسمها color موجودة داخل الكلاس Car في هذه الحالة نستطيع الوصول إلى الدالة color الموجودة داخل ال كلاس Car بواسطة الكائن c1 بالطريقة التالية

```
c1.colorr ();
```

•ماهو الكلاسclass

في هذا الدرس سوف نتحدث عن ماهو الكلاس class بشكل مختصر ومبسط

عندما تقوم ببناء برنامجك بشكل عادي فإنك حتماً سوف تقوم بتعريف متغيرات (variableالخصائص)

وأيضاً تقوم ببناء دوال) function الطرق)

في object oriented programmm كل ماعليك هو فقط أن تقوم بتعريف الدوال والمتغيرات داخل كلاس class

•لماذا نقوم بتعريف المتغيرات والدوال داخل كلاس class ؟

-الجواب لأن الكلاس class يوفر لنا تقنيات رائعة مثل

.1الكبسلة والتي تسمى باللغة الإنجليزية encapsulationnn وهو أن تضع الكود المكتوب داخل كتلة واحدة يمكنك الوصول إليه وإعادة إستخدامه

.2تحديد درجات الوصول (الحماية(

يمكنك بواسطة object oriented programmm تحديد درجات الوصول إلى بياناتك (المتغيرات والدوال) أي حمايتها من إعادة إستخدامها من قبل مستخدم آخر

وتحدد درجات الوصول بالشكل التالي

النوعprivat

e : عندما يتم تعريف متغير أو دالة من نوع privateee فإن ذلك يعني أن الوصول إلى هذا المتغير أو الدالة متاح من نفس الكلاس فقط

النوع : protected عندما يتم تعريف متغير أو دالة من نوع : protecteddd فإن هذا يعني أن الوصول لهذا المتغير أو الدالة متاح من نفس الكلاس ومن كلاس آخر فقط إذا كان معرف معه في نفس الحزمة أو في نفس البرنامج وهذا يعني أننا نستطيع بناء أكثر من كلاس في برنامج واحد أو في حزمة واحدة

النوع :public عندما يتم تعريف متغير أو دالة من نوع publiccc فإن هذا المتغير أو الدالة يكون متاح الوصول إليهما من نفس الكلاس أو من كلاس آخر من داخل أو خارج الحزمة

كل الدوال والمتغيرات التي تعرف مباشرة تحت اسم الكلاس تعتبر خاصة بالكلاس classs وتعطى الصلاحيات للكلاس في تحديد نوع الوصول إلى الدوال والمتغيرات الخاصة به كما ذكرنا مسبقاً

ماهو الكائن object•

تعريف الكائن بشكل مختصر

انت في برنامجك العادي تقوم ببناء دوال وتقوم بتعريف متغيرات داخل البرنامج في الاوبجكت اورينتد oopp هو نفس الشيء وهو انك تقوم ببناء دوال وتعريف متغيرات ولكن هذه الدوال والمتغيرات في الاوبجكت اورينتد oop تكون داخل كلاس class ولن نستطيع الوصول إلى الدوال أو المتغيرات داخل الكلاس إلا عن طريق إنشاء كائن (بغض النظر عن المتغيرات والدوال من نوع(static
ويعرف الكائن بأنه نسخة من الكلاس نستطيع من خلاله الوصول إلى الدوال والمتغيرات داخل الكلاس
فعلى سبيل المثال لو افترضنا أن لدينا كلاس اسمه Testtt واردنا الوصول إلى دالة معينة داخل هذا الكلاس في البداية نقوم بإنشاء كائن من الكلاس بالطريقة التالية

Test T=new Test ();

#لاحظ قمنا بإنشاء كائن من الكلاس Test اسمه T يمكنك إعطاء اي اسم للكائن حسبما تشاء
بعد إنشاء الكائن من الكلاس يمكنك الآن الوصول إلى الدوال والمتغيرات داخل الكلاس
لو افترضنا أن الكلاس Test يحتوي على دالة اسمها Adddd ونريد الوصول إلى هذه الدالة في هذه الحالة سوف نستخدم الكائن T الذي قمنا بانشاءه من الكلاس Test للوصول إلى هذه الدالة بالطريقة التالية

Test T = new Test ();
T.Addd ();

في درس اليوم سوف نتعلم كيفية الوصول إلى محتوى الكلاس (الدوال – المتغيرات)
#لاحظ في الصورة المرفقة مع الشرح لدينا class اسمه Anas
هذا الكلاس class يحتوي على متغيرين a,b وعلى دالة اسمها Get_Val تقوم بطباعة ناتج جمع المتغيرين a,b بعد إعطائهما قيمة في البرنامج الرئيسي main
السؤال هنا كيف نستطيع الوصول إلى المتغيرات والدوال الموجودة في الكلاس class والتحكم بها
#الجواب هو بإنشاء كائن والكائن ال object هو نسخة من الكلاس class
بواسطة الكائن نستطيع الوصول إلى المتغيرات والقيم الموجودة في الكلاس
في البرنامج الرئيسي قمنا بإنشاء كائن اسمه A من الكلاس Anas بالطريقة التالية

Anas A=new Anas();

وكلمة new تعني إنشاء نسخة من نفس الكلاس لهذا الكائن A
بعد ذلك إستطعنا الوصول إلى المتغيرات a,b بواسطة الكائن A وذلك بكتابة الكائن ثم نقطة ثم اسم المتغير أو الدالة الموجودة في الكلاس
بالشكل التالي

A.a=10;
A.b=20;

بعد تحديد قيم المتغيرات قمنا بإستدعاء الدالة Get_Val والتي ستقوم بطباعة ناتج جمع المتغيرين a,b وبالتأكيد سوف يكون الناتج 30 حسب القيم المسندة لهما في البرنامج الرئيسي

```java
package saad1;
public class Saad1 {
    int sum(int n1,int n2)
    {
        return(n1+n2);
    }
    public static void main(String[] args) {
        Saad1 s=new Saad1();
        int n=s.sum(5,3);
        System.out.println(n);
    }

}// عائلة البرمجة
```

#الدرس_الثاني_عشر

#الكائنات_الموجهة #constructor (دالة البناء)

في هذا الدرس سوف نتناول موضوع مهم في مواضيع عالم الكائنات الموجهة ال object oriented programs

#ماهو_ال constructor هو عبارة عن دالة تحمل نفس اسم الكلاس ويتم تنفيذه تلقائياً بمجرد إنشاء كائن من اسم الكلاس

مميزاته

- يتم تنفيذ ال constructor تلقائياً بمجرد إنشاء كائن من الكلاس
- يستخدم ال constructor لتهيئة المتغيرات وإعطاء قيم أولية واستدعاء الدوال التي نريد تنفيذها مع بدء تنفيذ البرنامج
- تستطيع بناءه بمعاملات أو بدون معاملات
- ال constructor لا يحتوي على نوع كبقية الدوال يعني لا يمكن أن نصنف constructor على أنه من نوع int أو من نوع void فهو لا يقبل النوع ولا يعيد قيمة لأن مهمته فقط العمل عند بمجرد إنشاء كائن لتهيئة بعض المتغيرات وإعطائها قيم أولية
- ال constructor يقبل إنشاءه أكثر من مرة بشرط أن يختلف كل constructor عن الأخر في عدد المعاملات وهذا مايسمى ب overload أي مجموعة من الدوال تحمل نفس الإسم ولكن تختلف في البارامتر

وال constructor الذي نتحدث عنه ليس خاص بالجافا فقط إنما هو عام في علم الكائنات الموجهة **object oriented programs**

مثال بسيط

#لاحظ في المثال التالي لدينا كلاس اسمه و saa2 ولدينا constructor يحتوي على امر طباعة جملة Welcom Iam constructor بمجرد أن نقوم بإنشاء كائن في البرنامج الرئيسي سيذهب تلقائياً لتنفذ constructor وسيقوم بطباعة جملة Welcom Iam constructor

```
public class Saad2}

   Saad2()
}
      System.out.println("Welcom Iam constructor" );
{
   public static void main(String[] args} (
```

```
        Saad2 s=new Saad2؛()
{
```

في المثال التالي لدينا برنامج يوضح فيه إمكانية إنشاء أكثر من constructor وهذا مايسمى overload constructor شكل من أشكال polymorphism ولكن بجب أن يختلفو في البارامترات ال constructor بدون بارامتر والثاني لديه 2 بارامترات والثالث لديه واحد بارامتر في البرنامج الرئيسي قمنا بإنشاء كائن عادي بدون بارامتر
بالشكل التالي

Saad2 s1=new Saad2();

في هذه الحالة سيتم إستدعاء ال constructor الأول الذي ليس لديه بارامترات
في السطر التالي قمنا بإنشاء كائن اسمه s2 مع اثنين بارامترات وهما العددين 3و5 في هذه الحالة سيتم إستدعاء ال constructor الثاني الذي لديه اثنين بارامترات أي ال constructor المطابق للبارامترات التي سيرسلها الكائن
وسيتم إسناد القيم المرسلة إلى المتغيرين x وy

Saad2 s2=new Saad2(5,3);

بعد ذلك قمنا بإنشاء كائن اسمه s3 مع بارامتر واحد بالشكل التالي

Saad2 s3=new Saad2(5);

وسيتم إسناد القيمة المرسلة إلى المتغيرين x وy

بعد ذلك قمنا بعملية الطباعة

System.out.println(s1.x + " " + s1.y);

في المرة الأولى سيطبع القيمة صفر للمتغيرين x و y لأن الكائن s1 لم يرسل أي قيم ولم يتم إسناد قيم معينة عند الإستدعاء سيقوم بطباعة النتيجة 0 للمتغيرين
في المرة الثانية قمنا بطباعة قيم ال x و y الخاصة بالكائن s2 سقوم بطباعة القيم 5,3 لأننا عند إنشاء الكائن s2 قمنا بإرسال عددين فاستدعى ال constructor المطابق وتم إسنادهما إلى المتغيرين x y ,
ستكون النتيجة 5,3
في المرة الأخيرة قمنا بطباعة قيم ال x و y الخاصة بالكائن s3 بالشكل التالي

System.out.println(s3.x + " " + s3.y);

وهنا سيتم طباعة القيمة 5

```
package saad2;

public class Saad2 {
     int x,y;
    Saad2()
    {
        System.out.println("Welcom Iam constructor");
    }
    Saad2(int a,int b)
    {
    x=a;
    y=b;

    }
    Saad2(int a)
    {
    x=a;
    y=a;
    System.out.println("x= " + x + "y= " + y);
    }
    public static void main(String[] args) {
        Saad2 s1=new Saad2();
```

```
        Saad2 s2=new Saad2(5,3);
        Saad2 s3=new Saad2(5);
        System.out.println(s1.x + "  " + s1.y);
        System.out.println(s2.x + "  " + s2.y);
        System.out.println(s3.x + "  " + s3.y);
    }

}
```

```
package saad2; // نعلم البرمجة من البداية حتى الإحتراف
public class Saad2 {
     int x,y;
    Saad2()
    {
        System.out.println("Welcom Iam constructor");
    }
    Saad2(int a,int b)
    {
    x=a;
    y=b;

    }
    Saad2(int a)
    {
    x=a;
    y=a;
    System.out.println("x= " + x + "y= " + y);
    }
    public static void main(String[] args) {
        Saad2 s1=new Saad2();
        Saad2 s2=new Saad2(5,3);
        Saad2 s3=new Saad2(5);
        System.out.println(s1.x + "  " + s1.y);
        System.out.println(s2.x + "  " + s2.y);
        System.out.println(s3.x + "  " + s3.y);
    }

}
```

الدرس الثالث عشر

الفصل الثاني البرمجة بالكائنات الموجهة object Oriented Programs

الكبسلة Encapsulation

1. **الكبسلة** والتي تسمى باللغة الإنجليزية**Encapsulation** وهو أن تضع الكود المكتوب داخل كتلة واحدة يمكنك الوصول إليه وإعادة إستخدامه
.2تحديد درجات الوصول)الحماية (
يمكنك بواسطة object oriented Program تحديد درجات الوصول إلى بياناتك (المتغيرات والدوال) أي حمايتها من إعادة إستخدامها من قبل مستخدم آخر أو كلاس أخر ويمكنك أعطاءه الصلاحيات بذلك أذا أردت
وتحدد درجات الوصول بالشكل التالي
النوع private
: eعندما يتم تعريف متغير أو دالة من نوع Private فإن ذلك يعني أن الوصول إلى هذا المتغير أو الدالة متاح من نفس الكلاس فقط
النوع : protected عندما يتم تعريف متغير أو دالة من نوع : protected فإن هذا يعني أن الوصول لهذا المتغير أو الدالة متاح من نفس الكلاس ومن كلاس آخر فقط إذا كان معرف معه في نفس الحزمة أو في نفس البرنامج وهذا يعني أننا نستطيع بناء أكثر من كلاس في برنامج واحد أو في حزمة واحدة
النوع :public عندما يتم تعريف متغير أو دالة من نوع public فإن هذا المتغير أو الدالة يكون متاح الوصول إليهما من نفس الكلاس أو من كلاس آخر من داخل أو خارج الحزمة
كل الدوال والمتغيرات التي تعرف مباشرة تحت اسم الكلاس تعتبر خاصة بالكلاس class وتعطى الصلاحيات للكلاس في تحديد نوع الوصول إلى الدوال والمتغيرات الخاصة به كما ذكرنا مسبقاً

في الصورة المرفقة مع الصورة

يوجد لدينا كلاسين

- الكلاس A1
- الكلاس B1

#لاحظ أن البرنامج الرئيسي يقع ضمن دالة الكلاس B1

في الكلاس A1 لدينا عدة متغيرات

- المتغير a وهو معرف من الدرجة private وهذا يعني أن الوصول إليه متاح فقط من الكلاس نفسه

في البرنامج الرئيسي قمنا بإنشاء كائن من الكلاس A1 وأردنا إسناد قيمة إلى المتغير a فلم نستطيع ذلك لأنه معرف من الفئة private وكما تلاحظ في الصورة فإن المترجم قد وضع لنا خط أحمر وهذا يعني أنه لا يمكننا الوصول إليه

- المتغير b وهو معرف من الفئة protected وهذا يعني أن الوصول إليه متاح ضمن الحزمة فقط والحزمة هي التي تضم داخلها مجموعة من الكلاسات وبما ان البرنامج يقع ضمن الحزمة نفسها إستطعنا الوصول إلى المتغير b وإسناد قيمة إليه
- المتغير c معرف من النوع public وهو متاح الوصول إليه من أي مكان سواءً من نفس الكلاس أو خارج الكلاس أو خارج الحزمة

```
package b1;// عائلة البرمجة

class A1
{
    private int a;
    protected int b;
    public int  c;

}
public class B1 {
    public static void main(String[] args) {
        A1 s=new A1();
        s.a=10; // private  خطأ لأنه محمي من نوع
        s.b=11;
        s.c=12;

    }

}
```

الدرس الرابع عشر

الفصل الثاني البرمجة بالكائنات الموجهة object Oriented Programs

الوصول إلى المتغيرات المحمية في كلاس أخر access to private variable in other calss

- تعلمنا في الدرس الثالث عشر تحديد درجات الوصول التي هي private وprotected و public
- تعلمنا ان المتغير الذي من نوع private لا يمكن الوصول إليه إلا إذا كنت تتبع نفس الكلاس

أولاً يجب أن تسأل نفسك مالذي أريده من هذا المتغير المحمي حتى أصل أليه

الجواب دائماً نريد أن نصل إلى المتغيرات المحمية في كلاس اخر لإحدى عمليتين

العملية الأولى هو تخزين قيمة في المتغير المحمي

العملية الثانية هو طباعة قيمة مخزنة داخل المتغير المحمي

في هذا الدرس سوف نتعلم كيفية الوصول إلى متغيرات محمية في كلاس آخر بواسطة دوال get و set

الدالة set يتم إستخدامها لتخزين قيمة في متغير محمي

الدالة get يتم استخدامها لطباعة قيمة من متغير محمي

تعرف هاتان الدالتان على انها من نوع public حتى تكون متاحة الوصول إليها من أي كلاس أخر .

#لاحظ في المثال الذي في الصورة المرفقة مع هذا الدرس يوجد لدينا كلاس B1 يحتوي على متغير محمي من نوع private اسمه price ونحن نحتاج هذا المتغير لأنه يحتوي على سعر البضاعة لطباعة سعر البضاعة على الشاشة ولكن طالما أننا نريد أن نطبع قيمته من البرنامج الرئيسي لن نستطيع ذلك لأن البرنامج الرئيسي يتبع الكلاس A1 ولا يتبع الكلاس B1 لذلك لن نستطيع الوصول إلى المتغير price في الكلاس B1 إلا بإستخدام دوال get و set الموجودة داخل الكلاس B1

نحن على سبيل المثال نريد طباعة اسم وسعر البضاعة

لا مشكلة لدينا في اسم البضاعة لأن متغير اسم البضاعة موجود في نفس الكلاس A1 وبما أن البرنامج الرئيسي موجود ضمن الكلاس A1 نستطيع الوصول إليه حتى ولو كان محمي

المشكلة التي نواجهها أننا نريد الوصول إلى متغير السعر price الموجود في الكلاس B1 وكما قلنا لن نستطيع الوصول إليه إلا بإستخدام الدوال get و set

في المرة الأولى نريد تخزين سعر البضاعة في المتغير price إذا نستخدم الدالة set التي بدورها ستقوم بإستقبال السعر ثم تخزينه في المتغير price وكأنها تقوم بدور الوسيط

في المرة الثانية نريد طباعة السعر سنستخدم أيضاً الدالة get والتي بدورها سوف تعيد لنا قيمة المتغير price

#لاحظ في البرنامج الرئيسي قمنا بإنشاء كائن من الكلاس A2 اسمه ob1 وكائن أخر من الكلاس B2 اسمه ob2

استطعنا الوصول إلى المتغير name من الكلاس A2 لماذا ؟ لأن البرنامج الرئيسي main يتبع الكلاس A1

قمنا بإعطاء قيمة للمتغير name وبعد ذلك أردنا الوصول إلى المتغير price الموجود داخل الكلاس B2 وهذا مستحيل لأنه private ونريد إستدعائه من خارج الكلاس لذلك لن نستطيع الوصول إليه إلا بإستخدام الدالتين get و set

في المرة ألأولى قمنا بإستخدام الدالة set لتخزين السعر في المتغير price والتي بدورها ستتقبل القيمة وتخزنها في المتغير price لأنها موجودة معه في نفس الكلاس

وفي المرة الثانية قمنا بطباعة البيانات وطبعنا السعر بإستخدام الدالة get

```java
package a2;// عائلة البرمجة
class B
{
    private int price;
     public void set(int x)
    {
    price=x;
    }
    public int get()
    {
    return price;
    }
}
public class A2 {
     private String name;
    public static void main(String[] args) {
        A2 ob1=new A2();
        B ob2=new B();
        ob1.name="Cable iphone7";
        ob2.set(20);
        System.out.println(ob1.name + " " + ob2.get());
    }

}
```

الدرس الخامس عشر

الفصل الثاني البرمجة بالكائنات الموجهة object Oriented Programs

- الوراثة inheritance

- كما هو في العالم الواقعي أن يرث الإبن بعض من صفات أبيه أو أمه مثل الطول – الحجم – لون الشعر

يجب التركيز على هذه النقطة (الولد يرث بعض الشبه من أبيه مع الإحتفاظ بخصائض تميزه عن أبيه)

- **في عالم object Oriented يمكن لكلاس ان يرث من كلاس أخر**

يسمى الكلاس الوارث الإبن والكلاس الذي تم التوريث منه يسمى الكلاس الأب

- يستطيع الكلاس الإبن الوصول إلى كافة المتغيرات والدوال الموجودة في الكلاس الأب ماعدا تلك الدوال والمتغيرات المحمية المعرفة من نوع private لأنه كما قلنا مسبقاً بأن الدوال والمتغيرات المحمية هي خاصة بالكلاس نفسه وللوصول إليها هناك طرق تكلمنا عنها في الدرس الرابع عشر

- ماهي الفائدة من الوراثة

بشكل عام نستفيد من الوراثة هو تنظيم وترتيب البرنامج واختصار الكود فإذا كان هناك متغير سوف نستخدمه في أكثر من كلاس يمكن تعريف هذا المتغير في الكلاس الأب حتى يمكن إعادة إستخدامه من كافة الكلاسات الأبناء

نفس الشيء إذا كان هناك دالة ونريد إستخدامها في أكثر من كلاس نكتفي بتعريفها في الكلاس الأب حتى يمكن إعادة إستخدامه من كافة الكلاسات الأبناء وبذلك تكون عرفت المتغير والدالة مرة واحدة وحررت مساحة في الذاكرة وحافظت على سرعة برنامجك وخفته

- مثال

لنفترض أن هناك مصنع سيارات يصنع أربعة أنواع من السيارات

نوع A نوع B نوع C نوع D

من الطبيعي أن هذه الأنواع تشترك في كثير من الصفات مثل

- اللون
- الحجم
- السرعة

ولكنها تختلف في الطراز فكل سيارة لها طرازها الخاص

في هذه الحالة هل سنقوم بتعريف متغيرات اللون والحجم والطراز والسرعة في كل كلاس !

طالما أن الكلاسات تشترك في هذه الصفات اللون والحجم والسرعة يمكننا أن نعرفها في كلاس يسمى الكلاس الأب ونجعل بقية الكلاسات يرثون هذه الصفات من الكلاس الأب حتى يستطيعون إستخدام متغيرات اللون والحجم والسرعة وبذلك نكون عرفنا هذه المتغيرات اللون والحجم والسرعة مرة واحدة بدلاً من تعريفها أكثر من مرة وبهذا نكون قد قللنا من عدد المتغيرات وحررنا مساحة في الذاكرة وساعدنا على سرعة وخفة نظامنا وهذه من أهم فوائد الوراثة

- طريقة الوراثة في جافا

إذا أردنا من الكلاس B أن يرث من الكلاس A نستخدم الأمر التالي

Class B extends A

الآن يستطيع الكلاس B الوصول إلى كافة المتغيرات والدوال الموجودة في الكلاس A وإعادة إستخدامها ماعدا تلك الدوال والمتغيرات المعرفة من نوع private كما تحدثنا سابقاً

في البرنامج التالي لدينا التالي

1- كلاس car وهو الكلاس الأب ويحتوي على المتغيرات التالية

```
int size,speed;
String color;
```

قمنا بتعريف هذه المتغيرات في الكلاس الأب لأننا سنحتاج إعادة إستخدامها في بقية الكلاسات الأبناء

2- كلاس اسمه Car_Model_A وهو يرث من الكلاس الأب car بواسطة أمر الوراثة extends بالشكل التالي

Car_Model_A extends car

3- لدينا كلاس اسمه Car_Model_B وهو يرث من الكلاس car بنفس الطريقة التي ورث منها الكلاس Car_Model_A

استفدنا من عملية الوراثة اننا قمنا بتعريف المتغيرات المشتركة في الكلاس الأب car وبذلك عرفناها مرة واحدة واختصرنا الجهد والكود وحررنا مساحة أكثر في الذاكرة لان كل متغير نقوم بتعريفه يقوم بحجز مساحة في الذاكرة

في البرنامج الرئيسي قمنا بإنشاء كائن object من الكلاس Car_Model_A اسمه A بالطريقة التالية

```
Car_Model_A A=new Car_Model_A();
```

واستطعنا بواسطة هذا الكائن الوصول إلى المتغيرات الموجودة في الكلاس الأب car

بعد ذلك قمنا بإسناد قيم إلى المتغيرات

```
A.name="EE1454"; A.color="Red";A.size=400;A.speed=380;
```

وبنفس الطريقة قمنا بإنشاء كائن من الكلاس الأخر اسمه B بالطريقة التالية

```
Car_Model_B B=new Car_Model_B();
```

واستطعنا بواسطة هذا الكائن الوصول إلى المتغيرات الموجودة في الكلاس الأب car

- بعد ذلك تم إسناد قيم إلى المتغيرات الموجودة في الكلاس الأب والوصول إليها مباشرة

ثم إستدعاء دالة الطباعة الموجودة في الكلاس لتطبع البيانات التي قمنا بإسنادها إلى المتغيرات الموجودة في الكلاس الأب

البرنامج كاملاً في الصفحة التالية

```
package car;
class car
{
    int size,speed;
    String color;
}
class Car_Model_A extends car
{
    String name;
    void showDetail()
    {
        System.out.println(name+ " " + color + " " + size + "  " + speed);
    }}
  public class Car_Model_B extends car
{
  String name;
    void showDetail()
    {
        System.out.println(name+ " " + color + " " + size + "  " + speed);
    }
    public static void main(String[] args) {
       Car_Model_A A=new Car_Model_A();
       Car_Model_B B=new Car_Model_B();
       A.name="EE1454"; A.color="Red";A.size=400;A.speed=380;
       B.name="BB1600"; B.color="Black";B.size=350;B.speed=280;
        A.showDetail(); B.showDetail();
    }

}
```

الدرس السادس عشر

الوراثة الهرمية في الجافا متعددة المراحل

هناك أوجه متعددة للوراثة المتعددة في الجافا سنأخذ اليوم أحد الأوجه
وهي الوراثة بطريقة الجد – الأب – الحفيد
لنفترض أن لدينا كلاس اسمه A ولدينا كلاس اسمه B
في حال إذا ورث الكلاس B من الكلاس A سيصبح ال A هو الأب للكلاس B
لنفترض أن لدينا كلاس ثالث اسمه C وهذا الكلاس C يرث من الكلاس B ستكون العلاقة كالتالي
الكلاس A هو الأب للكلاس B والكلاس B هو الأب للكلاس C
ستكون أيضاً علاقة A بالكلاس C بأن الكلاس A هو الجد للكلاس C
ولكن هل يستيطع الكلاس C أن يصل إلى محتويات (المتغيرات والدوال) الخاصة بالكلاس A
الجواب "نعم"
طالما أن الكلاس يرث من الكلاس B والكلاس B يرث من الكلاس A فإن بإستطاعة الكلاس C أن يصل إلى المتغيرات والدوال الخاصة للكلاس A ماعدا المتغيرات المحمية من النوع Private كما ذكرنا في الدرس الخامس عشر
في هذا المثال لدينا كلاس اسمه A
وكلاس اسمه B يرث من الكلاس A
وكلاس اخر اسمه C يرث من B
يستطيع الكلاس C الوصول إلى متغيرات الأب B ومتغيرات الجد C كما هو موضح في المثال
السؤال الأهم هل يستطيع الكلاس الأب الوصول إلى متغيرات ودوال الكلاس الإبن هذا ماسنشرحه في الدرس المقبل

```
package c;
class A
{
   int varA;
}
class B extends A
{
   int varB;
}
public class C extends B
{
   public static void main(String[] args) {
      C ob=new C();
      ob.varA=10;
      ob.varB=20;
      System.out.println(ob.varA);
       System.out.println(ob.varB);
   }}
```

الدرس السابع عشر

ال Overriding

في الدروس السابقة تحدثنا عن الوراثة وقلنا أنه يمكن للكلاس الأبن أن يرث الكلاس الأب **super class** وبذلك يمكنه الوصول إلى الدوال والمتغيرات الموجودة في الكلاس الأب في هذا الدرس سوف نتحدث عن

المقصود بـ **Overriding** هو إمكانية تسمية أكثر من دالة بنفس الإسم
بحيث نستطيع أن نكتب دالة بنفس الإسم وبنفس البارامتر وبنفس النوع في الكلاس الأب والإبن بحيث يستطيع الأب الوصول إلى هذه الدوال الموجودة في الأبناء شريطة أن تكون بنفس إسم الدالة التي يمتلكها وهذا المقصود بـ **Overriding** بحيث يمكننا إعادة إستخدام الدالة الموجودة في الأب بنفس الإسم ونفس التركيبة في الكلاس الإبن ولكن عملها يختلف يعني فقط اختلاف في الكود

لنفترض أن لدينا في الكلاس دالة اسمها **calc** عمل هذه الدالة جمع عددين
نستطيع إستنساخ الدالة **calc** في الكلاس الأبن بنفس الإسم ونفس التركيبة ونجعلها للطرح
في المثال المرفق مع المنشور نوضح كيف يستطيع الأب الوصول إلى دالة موجودة في الإبن شريطة أن تكون الدالة التي الكلاس الإبن بنفس الإسم
في البرنامج الرئيسي قمنا بإشتقاق كائن من الكلاس الأب يشير إلى الكلاس ألأب بالطريقة التالية

A ob1=new A();

في هذه الحالة سوف يستدعي الدالة **calc** الموجودة في الكلاس الأب
بعد ذلك قمنا بإشتقاق كائن من الكلاس الأب يشير إلى الكلاس الإبن بالطريقة التالية

A ob2=new B();

في هذه الحالة سوف يتم إستدعاء الدالة **calc** الموجودة في الكلاس الإبن

```
package c;
class A //
{
   void calc(int a,int b)
   {
      System.out.println(a+b);
   }
}
class B extends A
{
   void calc(int a,int b)
   {
      System.out.println(a-b);
   }
}
public class C extends B
{
   public static void main(String[] args) {
   A ob1=new A();
   A ob2=new B();
   ob1.calc(5, 10);
   ob2.calc(5, 10);
   }}
```

الدرس الثامن عشر

- **الأمر final في جافا**

Final Keyword In Java

يستخدم الأمر final في جافا لتقييد المستخدم بقيمة معينة ثابتة لا يمكن تجاوزها أو تغييرها فعلى سبيل المثال إذا عرفنا متغير بأنه final بالطريقة التالية

```
final int n=90;
```

هنا سيصبح قيمة المتغير n قيمته 90 ولا يمكننا تغييرها فعلى سبيل المثال لو أردنا تعديل قيمة المتغير n لن نستطيع ذلك

يمكننا أيضاً تعريف الدالة بأنها final بالشكل التالي

```
class A{
    final void show(){System.out.println("Hello");}
}
```

1. **وعند تعريف الدالة بأنها من النوع final لا يمكن تطبيق علها ال override والذي تعلمناه في الدرس السابق**
2. **#المثال التالي يوضح انه لا يمكن تطبيق طريقة override على دالة من نوع final**

```
class A{
  final void show(){System.out.println("Hello A");}
}

class B extends A{
   void run(){System.out.println("Hello B");}

   public static void main(String args[]){
   B ob1= new B();
   Ob1.run();
   }
}
```

في الصورة المرفقة مع المثال توضح أنه لا يمكننا أن نستطيع أن نجعل الكلاس B يرث من الكلاس A لأن الكلاس A معرف من النوع final وبالتالي لا يمكن لبقية الكلاسات أن ترث منه لاحظ وجود الخط الأحمر على اسم الكلاس لأننا أردنا تطبيق عملية الوراقة وهذا يعني أنه لا ينكن للكلاس B أن يرث من الكلاس A وايضاً وجود خط أحمر على اسم الدالة show في على الدالة Show الموجودة في الكلاس B لأننا لا يمكننا تطبيق مبدأ ال Overriding لأن الدالة في الكلاس A معرفة بأنها من النوع final أيضاً

البرنامج في الصفحة التالية

```java
package c;
 final class A
{
     final void Show()
     {
         System.out.println("Hello");
     }
}
class B extends A
{
     void Show()
     {
          System.out.println("Hello");
     }
}
public class C
{
     public static void main(String[] args) {
   B ob1=new B();
   ob1.Show();

     }}
```

الدرس التاسع عشر object oriented java

- ال Super

super keyword in java

ال super في الجافا هي كلمة محجوزة تشير إلى الكلاس الأب ومتغيرات ودوال الكلاس الأب

- إستخدامات الكلمة super في الجافا

1- تستخدم الكلمة super لإستدعاء متغيرات الكلاس الأب
2- تستخدم الكلمة super لإستدعاء دوال المتغير الأب بطريقة فورية
3- يمكن إستخدام ال super لإستدعاء دوال البناء في الكلاس الأب

- مثال لإستخدام الكلمة المحجوزة super لإستدعاء متغيرات الكلاس الأب

```
class A{
String color="My Color is Green";
}
class B extends A{
String color="My Color is Red";
void printColor(){
System.out.println(color);//prints color B class
System.out.println(super.color);//prints color of A cl
ass
}
}
class TestSuper1{
public static void main(String args[]){
B d=new B();
d.printColor();
}}
```

في المثال السابق تم تعريف calss اسمه A ولديه متغير اسمه color يحتوي على الجملة "My Color is Green";

لدينا أيضاً كلاس اسمه B يرث من الكلاس A

الكلاس B لديه ايضاً متغير اسمه color يحتوي على الجملة التالية

"My Color is Red

الكلاس B يحتوي على دالة اسمها printColor تقوم بطباعة المتغير color الخاص بالكلاس B وفي نفس الدالة استخدمنا الكلمة super.color وبهذه الطريقة ستقوم الدالة printColor بطباعة قيمة المتغير color الخاص بالكلاس B ثم طباعة قيمة المتغير color الخاص بالكلاس A لأننا استخدمنا الكلمة المحجوزة super

- مثال لإستدعاء دوال الأب بواسطة الكلمة المححجوزة super

``class A{ String color="My Color is Green"; void printColor(){ System.out.println(color); } } class B extends A{ String color="My Color is Red"; void printColor(){ System.out.println(color); super. printColor(); } } class TestSuper1{ public static void main(String args[]){ B d=new B(); d.printColor(); }}```	سنستخدم نفس المثال السابق مع إضافة دالة اسمها printcolor في الكلاس الأب ثم نستدعيها بواسطة الكلمة المحجوزة super في الكلاس B في نفس الكلاس B توجد دالة اسمها PrintColor تقوم بطباعة المتغير ثم يتم إستدعاء دالة الأب PrintColor الخاصة بالكلاس الأب عن طريق الكلمة المحجوزة super

مثال لإستدعاء دالة البناء Constructor الخاصة بالأب super

في المثال التالي لدينا الكلاس A يحتوي على دالة بناء في هذه الدالة يتم طباعة قيمة المتغير color
في الكلاس B الإبن نقوم بإستدعاء دالة البناء الخاصة بالكلاس الأب بواسطة الأمر super()
خلاصة الدرس انه إذا كتبنا الكلمة المحجوزة لوحدها super() تقوم بإستدعاء دالة البناء الخاصة بالأب Constructor
وإذا أردنا إستدعاء متغير أو دالة خاصة بالأب نقوم بكتابة الكلمة super ثم نقطة ثم كتابة اسم المتغير او الدالة

```
Super.varibleName;
Super.FunctionName();
```

```
class A{
String color="My Color is Green";
A (){
System.out.println(color);
}}
class B extends A{
String color="My Color is Red";
void B (){
Super();
System.out.println(color);
} }
class TestSuper1{
public static void main(String args[]){
B d=new B();
d.printColor();  }}
```

الدرس العشرون تعدد الأشكال في الجافا
Polymorphism in Java

تعدد الأشكال في جافا هو مفهوم يمكننا من خلاله القيام بعمل واحد بطرق مختلفة
ويجب ان تلاحظ كلمة أشكال وليس أشكال أي ان هناك أشكال عدة وليس شكل واحد
والشكل الأكثر شهرة هو أن نجعل كائن مشتق من الأب يشير إلى الكلاسات الأبناء ويستطيع الوصول إليها وإعادة استخدامها
وأشهر أستخدامات تعدد الأشكال هو إمكانية وصول الأب إلى الدوال والمتغيرات الخاصة بالكلاس الإبن وإعادة إستخدامها ويشترط لوصول الأب إلى دوال الكلاس الإبن أن تكون الدوال على مبدأ ال override أي أن الدالة التي يستطيع الأب أن يصل إليها تكون بنفس إسم الدالة الموجودة في الأب
كيف يستطيع الكلاس الأب الوصول إلى متغيرات ودوال الكلاس الإبن
الإجابة نستطيع ذلك بواسطة مايسمى ال upcasting والمقصود بـ upcasting هو إنشاء كائن object من الكلاس الأب يشير إلى الكلاس الإبن بالطريقة التالية

```
class A{}
class B extends A{}
A a=new B();//upcasting
```

في الكود السابق تم توضيح طريقة upcasting وهو إنشاء كائن من الكلاس الأب يشير إلى الكلاس الإبن وذلك حتى يستطيع الكلاس الأب الوصول إلى متغيرات ودوال الكلاس الإبن

تعدد الاشكال وقت التشغيل بإستخدام ال override : ونقصد به هنا هو أن الأب يستطيع الوصول إلى دالة من دوال الكلاس الإبن في حالة إذا كانت هذه الدالة تطبق مبدء ال override لاحظ المثال التالي
تعدد الأشكال Polymorphism مع الدوال

```
class A{
  void run(){System.out.println("Iam Parent Class");}
}
class B extends A{
  void run(){System.out.println("Iam Child Class");}

  public static void main(String args[]){
    A b = new B();//upcasting
    b.run();
  }
}
```

في المثال السابق تم إنشاء كلاس اسمه A وكلاس اسمه B يرث من الكلاس A الآن يستطيع الكلاس الأب الوصول إلى الدالة الخاصة باكلاس الإبن بواسطة إنشاء كائن من الكلاس الأب يشير إلى الكلاس الإبن ومن ثم إستدعاء الدالة run الخاصة بالكلاس الإبن لأنها تطبق مبدء ال override أي ان الدالة run في الكلاس الإبن موجودة أيضاً في الكلاس الأب بنفس البنية لهذا استطاع الكلاس الأب الوصول إلى الدالة run فلو كانت الدالة run تحمل إسماً مختلفاً عن الدالة التي موجودة في الكلاس الأب لن يستطيع الأب الوصول إليها

تعدد الأشكال Polymorphism مع المتغيرات
يستطيع الكلاس الأب الوصول إلى المتغيرات الخاصة بالكلاس الإبن بشرط أن يكون المتغير الذي يصل إليه يحمل نفس اسم المتغيرات في الكلاس الأب المثال التالي يوضح ذلك

```
class A
{
double d=0.55;
```

```
}
public class A4 extends A{

double d=0.5;
public static void main(String[] args) {
A ob1=new A4();
System.out.println(ob1.d);
}

}
```

في المثال السابق لدينا الكلاس A الأب والكلاس A4 الإبن
#لاحظ في البرنامج الرئيسي قمنا بإنشاء كائن من الكلاس الأب A يشير إلى الإبن ثم استطاع الأب الوصول إلى المتغير d الموجود في الكلاس الإبن لأنه يحمل نفس اسم المتغير الموجود في الكلاس الأب ولو كان المتغير يحمل إسماً مختلفاً لن يستطيع الكلاس الأب الوصول إليه

```
class A
{
    double d=0.55;
    void run() {System.out.println("Welcome Iam A");}
}
public class A4 extends A{

   double d=0.5;
   double d1=0.9;
   void run() {System.out.println("Welcome Iam A4");}
    void runTime() {System.out.println("Welcome Iam A4");}
     public static void main(String[] args) {
        A ob1=new A4();
        System.out.println(ob1.d);
        System.out.println(ob1.d1);
        ob1.run();
        ob1.runTime();
     }


}
```

في المثال السابق
لدينا كلاس اسمه A وهو الأب ولدينا كلاس اسمه A4 وهو الإبن الذي يرث من الأب A في البرنامج الرئيسي قمنا بإنشاء كائن من الأب يشير إلى الإبن بالطريقة التالية

A ob1=new A4();

بعد ذلك استطعنا الوصول إلى المتغير d الموجود في الكلاس الإبن بواسطة الكلاس الأب لأن المتغير d موجود في الكلاس الأب بنفس النوع
الأب لا يستطيع الوصول إلى المتغير d1 لأنه لا يوجد متغير اسمه d1 في الكلاس الأب بنفس الطريقة استطاع الأب الوصول إلى الدالة run لأنها موجودة أيضاً في الكلاس الأب وبنفس البنية بينما لم يستطيع الوصول إلى الدالة runtime لأنها غير موجودة في الكلاس الأب

الدرس الواحد والعشرون الكلمة المحجوزة this

الكلمة المحجوزة this في جافا

This هي كلمة محجوزة تشير إلى الكائن أو الكلاس الحالي

- **إستخدامات الكلمة المحجوزة this**

1- بواسطة this نستطيع أن نشير إلى متغيرات الكلاس الحالي
2- بواسطة this نستطيع إستدعاء دوال الكلاس الحالي
3- بواسطة this نستطيع إستدعاء دالة البناء constructor الخاصة بالكلاس
4- بواسطة this نستطيع إستدعاء دالة وتمرير المعاملات إليها
5- بواسطة this نستطيع تمرير إلى دالة البناء constructor وتمرير معاملات إليها

- **إستخدام الكلمة المحجوزة this للإشارة إلى متغيرات الكلاس الحالي**

في المثال المقابل يوجد لدينا كلاس يحتوي على المتغير x والمتغير y ولدينا دالة لديها متغيرين كمعاملات بنفس اسم المتغيرات
في هذه الحالة سوف نقابل مشكلة وهو ان المفسر لن يستطيع التمييز بين ال x و y الخاص بمعاملات الدالة وبين ال x و y الحاص بالكلاس

```
class A {
 int x,y;
 void setvalue(int x,int y)
 {
    x=x;
    y=y;
 }
}
```

لحل المشكلة نقوم بإستخدام الكلمة المحجوزة this للإشارة إلى متغيرات الكلاس وهنا سيستطيع المفسر ان يميز بين متغيرات الكلاس ومتغيرات الدالة شاهد الشكل المقابل
هنا سيتم تخزين قيمة ال x وقيمة ال y التي تم تمريرها إلى الدالة setvalue إلى المتغيرات x و y الخاصة بمتغيرات الكلاس

```
class A {
 int x,y;
 void setvalue(int x,int y)
 {
    this.x=x;
    this.y=y;
 }
}
```

في الشكل المقابل لا نحتاج إلى الكلمة المحجوزة this لأن معاملات الدالة setvlaue تختلف عن أسماء متغيرات الكلاس

```
class A {
 int x,y;
 void setvalue(int r,int c)
 {
    this.x=r;
    this.y=c;
 }


}
```

- **إستخدام الكلمة المحجوزة this لإستدعاء دوال الكلاس الحالي**

في المثال المقابل إستطعنا إستدعاء showvlue() بواسطة الكلمة المحجوزة this

```
class A {
 int x,y;
  void showvlaue()
 {
   System.out.println("x= "+ x + " y = "+ y);
 }
 void setvalue(int x,int y)
 {
    this.x=x;
    this.y=y;
    this.showvlaue();}
```

- **إستخدام الكلمة المحجوزة this لإستدعاء دوال البناء Constriction للكلاس الحالي**

```
class A {
 A()
{
 System.out.println("Iam Class A");

 }
A(int x,int y)
{
   this();
   System.out.println(x+y);
}



public static void main(String[] args)
{
 A a=new A(10,5);


}}
```

#لاحظ في المثال المقابل تم استدعاء دالة البناء A() بواسطة this والمشار إليها باللون الأصفر

```
class A {
 A()
{
 System.out.println("Iam Class A");
     this(5,10);

 }
A(int x,int y)
{
   System.out.println(x+y);
}
public static void main(String[] args)
{
 A a=new A();
}}
```

#لاحظ في المثال المقابل تم إستدعاء دالة البناء بواسطة الكلمة المحجوزة this مع تمرير معاملات

إستخدام الكلمة المحجوزة this لإرجاع كافة متغيرات ودوال الكلاس

```
class A
{
 A GetA()
{
  return this;

 }
```

الشكل المقابل يوضح كيفية بناء دالة ويكون المرتجع هو كائن من نوع الكلاس وبهذا نستطيع من خلال هذه الدالة الوصول إلى كافة متغيرات ودوال الكلاس مثل الكائن تماماً بشكل سريع عند استدعاء الكلاس

بقية المثال في الأسفل يوضح كيفية الوصول إلى بقية دوال ومتغيرات الكلاس مباشرة بدون إنشاء كائن وذلك لأنه قمنا بإرجاع الكائن مباشرة في الدالة GetA التي هي من نوع كلاس فتعاملنا معها كما نتعامل مع الكائن

```
package c;
 class A
{
  A GetA()
 {
   return this;

  }
 void SHowmessage()
 {
    System.out.println("Iam Class A");
 }
 public static void main(String[] args)
 {
 new A().GetA().SHowmessage();
 }}
```

الدرس الثاني والعشرون الكلاس المجرد abstract

Abstraction in Java

التجريد في الجافا
كلمة abstract هي كلمة محجوزة في الجافا تعني أن الكلاس أو الدالة التي يتم تعريفها على أنها abstract تكون مجردة
مامعنى التجريد : عندما يتم الإعلان عن كلاس على انه من نوع abstract أي مجرد فإن هذا الكلاس يستخدم فقط للتصريح عن الدوال فقط بدون بناء كود داخل هذه الدالة وأي كلاس أخر يرث من الكلاس المجرد الذي من نوع abstract يجب على الكلاس الوارث إلزامياً تنفيذ كافة الدوال التي تم الإعلان عنها في الكلاس المجرد
#مثال

```
abstract class A
{
  abstract void show();
}
```

#لاحظ تم تعريف الكلاس A على أنه كلاس مجرد من نوع abstract وسمي مجرد لأنه لن يسمح لك ببناء كود داخله فقط سيسمح لك بالتصريح أو الإعلان عن دوال ومتغيرات
إذا قمنا ببناء كود داخل دالة مصرحة فإن الكلاس المجرد abstract لن يسمح بذلك

```
abstract class A
{
   abstract  void run(){System.out.println("Hello");}
   abstract void print();
}
```

في الشكل السابق سوف تحصل على رسالة خطأ من المترجم لأن الكلاس المجرد الذي من نوع abstract يسمح لك فقط بالتصريح عن الدوال فقط ولا يسمح لك ببناء كود داخل الدالة
#ماهي_الفائدة_من_التجريد
الفائدة من التجريد هو إيجاد **نمط مرسوم**
بمعنى أننا نجبر كل الكلاسات الوارثة على تنفيذ دوال بشكل إجباري على سبيل المثال
لنفترض ان لدينا كلاسين كالتالي

Class carA , Class carB

هذين الكلاسين عملهم طباعة مواصفات نوع السيارة التي تنتمي إلى الكلاس
لنفترض أننا نريد إجبار الكلاسين على طباعة لون السيارة ومقدار السرعة
في هذه الحالة سنقوم ببناء كلاس مجرد من نوع abstract ونقوم بالتصريح عن دالة اللون ودالة السرعة داخل الكلاس المجرد ثم نقوم بجعل الكلاسين carA و الكلاس carB يرثان من الكلاس المجرد
المثال التالي يوضح ذلك

```
abstract class car
{
  abstract void color();
  abstract void speed();
}
class carA extends car
{
   void color()
{System.out.println("The Color is Black");}
void speed ()
{System.out.println("the speed is 2000 Kh");}
}
class carB extends car
{
   void color()
{System.out.println("The Color is Black");}
void speed ()
{System.out.println("the speed is 2400 Kh");}
}
```

كما تلاحظ يتم تنفيذ الدالتين color والدالة speed إجبارياً للكلاسات الوارثة وهذا مانتسفيده من التجريد **رسم النمط** أو **تنفيذ خطة** بأن يلتزم كل كلاس وارث أن يقوم بتنفيذ دوالة مهمة في البرنامج لتلافي مشكلة النسيان

في البرنامج المقابل كود يوضح طريقة إنشاء كلاس مجرد abstract والتصريح عن دوال داخل هذا الكلاس ونشاهد أيضاً كيف أن الكلاس الوارث يقوم بتنفيذ الدوال المصرح عنها في الكلاس الأب إجبارياً

```
package a1;
abstract class A
{
   abstract  void run();
   abstract void print();
}
public class A1 extends A{
   void run()
   {
      System.out.println("Hello");
   }
   void print()
   {
      System.out.println("Hello");
   }
   public static void main(String[] args) {
      A1 ob=new A1();
      ob.print();
      ob.run();
   } }
```

الدرس الثالث والعشرون

ال Interface في الجافا

ال interface ويسمى باللغة العربية الواجهة هو شبيه من حيث البنية بالكلاس ويحتوي على دوال مجردة من نوع abstract

ذكرنا في الدرس السابق ان كلمة مجرد أو abstract تعني ان الدالة أو الكلاس الذي من نوع abstract أنه لا يحتوي كود أي يستخدم للتصريح فقط

وال interface هو شبيه بالكلاس المجرد أي انه يحتوي على دوال مجردة من نوع abstract

ويطلق على ال interface في بعض المراجع مصطلح نمط أي ان الكلاس الذي سيرث من interface هو ملزم بتنفيذ كافة الدوال الموجودة داخل ال interface وهو شبيه بالكلاس من نوع abstract سوى أن الفرق بين ال interface يدعم الوراثة المتعددة

بعض الفروقات بين ال interface والكلاس abstract

1- يمكن أن يحتوي الكلاس من نوع abstract على دوال مجردة وغير مجردة بينما ال interface يحتوي على دوال مجردة فقط
2- الكلاس من نوع abstract لا يدعم الوراثة المتعددة كما هو الحال في بقية الكلاسات بينما ال interface يدعم الوراثة المتعددة
3- الكلاس من نوع abstract يمكن أن يحتوي على متغيرات ثابتة ومتغيرة بينما ال interface يحتوي على متغيرات ثابتة فقط
4- الكلاس من نوع abstract يمكن أن يرث من ال interface والعكس غير صحيح
5- الكلمة المحجوزة abstract تستخدم للكلاس والدوال بينما كلمة interface تستخدم للواجهة فقط

بعد مشاهدة الفروقات بين ال interface وال abstract نلاحظ أن ال interface يحتوي على التالي

1- دوال مجردة فقط
2- متغيرات ثابتة فقط
3- يجب على الكلاس الوارث أن يقوم بتنفيذ كافة الدوال الموجودة في ال interface

وهذا يجعلنا نفهم ان ال interface يتحكم بمسار البرنامج لذلك يطلق عليه في بعض المراجع مصطلح **نمط أو المسار المرسوم** بحيث أنه يقيد البرنامج بمسار معين وهو ان كل الكلاسات الوارثة من interface تقوم بتنفيذ كافة الدوال في ال interface والمتغيرات الثابتة

- طريقة وراثة الكلاس من ال interface

تعلمنا في الدروس السابقة ان الكلاس عندما يريد أن يرث من كلاس أخر يستخدم الكلمة Extends ولكن عندما يريد كلاس ان يرث من interface فإننا نستخدم الكلمة المحجوزة **implements** الطريقة التالية توضح كيف يرث كلاس من interface

```
class A implements P
```

في الطريقة السابقة وضحنا كيف يرث كلاس اسمه A من interface سامه p بواسطة الكلمة المحجوزة implements و implements تعني تنفيذ

- طريقة بناء ال interface

لبناء ال interface نكتب أولا الكلمة المحجوزة interface ثم اسم ال interface بالطريقة التالية

```
interface A
{

}
```

في الشكل المقابل تم بناء interface اسمه A ثم قوس مفتوح وقوس مغلق لتحديد جسم البرنامج أو هيكل البرنامج تماماً كما نبني الكلاس

- الطريقة التالية توضح كيف يرث كلاس من interface

```
interface A
{

}
public class Saad3 implements  A {


   public static void main(String[]
args) {

   }

}
```

في البرنامج المقابل قمنا ببناء interface اسمه A ثم وضحنا كيف قام الكلاس Saad3 بوراثة ال interface بواسطة الكلمة المحجوزة interface

```
interface A
{
  public int x=10؛
}
public class Saad3 implements  A}


    public static void main(String[] args} (
       Saad3 ob1=new Saad3؛()

       System.out.println(ob1.x);
}
```

كما ذكرنا في بداية الشرح أن ال interface لايقبل سوى متغيرات نهائية فقط أي أن لها قيمة نهائية لا يمكن تعديلها من الكلاسات الأخرى التي ترث من ال interface

```
interface A
{
  public int x=10;
  void print();
}
public class Saad3 implements  A {

    public void print()
    {
    System.out.println("Hello");
    }

    public static void main(String[] args) {
      Saad3 ob1=new Saad3();

      System.out.println(ob1.x);
      ob1.print();
    }

}
```

في البرنامج المقابل نستطيع طباعة قيمة ال x من الكائن الذي يتبع الكلاس الوارث Saad3 ولكن ال interface لايسمح بتعديل قيمة ال x على جميع الكلاسات الوارثة أن تتقيد بما يمليه عليها ال interface

كما أن الكلاس الوارث من interface ملزم بتنفيذ كافة الدوال المصرحة في interface كما يوضح الشكل التالي

```
package sbv;

interface Bank
{
    float rateOfprof();
}
 class QNN implements Bank{
    public float rateOfprof()
    {return 9.3f;}
}

public class SBV implements Bank{
    public float rateOfprof()
    {return 9.15f;}

    public static void main(String[] args) {
       Bank ob1=new QNN();
       Bank ob2=new SBV();

System.out.println(ob1.rateOfprof());

System.out.println(ob2.rateOfprof());
    }

}
```

في البرنامج المقابل يوضح كيفية تحقيق مبدء ال override مع ال interface وهو أن الأب ال interface يستطيع الوصول إلى الدوال الخاصة بالكلاسات الأبناء الوارثة منه وإمكانية إنشاء كائنات من ال interface

نكتفي بهذا القدر من الشرح وللشرح بقية

الدرس الرابع والعشرون

الوراثة المتعددة بواسطة ال interface
الجافا لا تدعم الوراثة إلا مع ال interface فمن مميزات ال interface أنه يدعم الوراثة المتعددة ويمكن للكلاس الواحد أن يرث من أكثر من interface
لنفترض أن اثنين interface الأول اسمه A والثاني اسمه B ولدينا كلاس اسمه Saad3 يستطيع أن يرث من interface A و interface B بالطريقة التالية

public class Saad3 implements A, B

'

في البرنامج التالي لدينا interface **اسمه A و** interface أخر اسمه B ولدينا كلاس اسمه Saad3 وفيه يتم توضيح طريقة الوراثة المتعددة وبذلك أصبح الكلاس Saad3 مُلزماً بتنفيذ كافة الدوال المصرحة في كلا ال interface

```
package saad3;
interface A
{

  void print();
}
interface B
{

  void msg();
}
public class Saad3 implements  A, B {
    public void print()
      { System.out.println("you implements interface A"); }
      public void msg()
     { System.out.println("you implements interface B");   }

    public static void main(String[] args) {
      Saad3 ob1=new Saad3();
      ob1.print();
      ob1.msg();
    }}
```

الفصل الثالث

برمجة الواجهات بإستخدام مكتبة swing

مكتبة swing هي المكتبة الرسومية التي تزود المستخدم بالعناصر اللازمة لبرمجة الواجهات مثل الإطارات Frame والأزرار Button الخ

في هذا الدرس بعون الله سوف نتعلم كيفية إنشاء واجهة فقط بإستخدام الكلاس Jframe

يستخدم الكلاس JFrame لإنشاء مايسمى بالواجهة أو الحاوية وهي الإطار الذي يضم كافة العناصر التفاعلية التي يتعامل معها المستخدم مثل الأزرار ومربعات النصوص والعناوين وهكذا

وتبسيطاً للمبتديء سوف نتعلم في هذ الدرس كيفية إنشاء إطار أو مايسمى بحاوية فقط دون التطرق إلى عناصر أخرى فقط حتى تكون دروسنا خطوة بخطوة

أولاً للبدء في برمجة الواجهات لا بد من تضمين مكتبة swing بالطريقة التالية

```
import javax.swing.*;
```

ثانياً لإنشاء واجهة المستخدم نستخدم الكلاس JFrame بالطريقة التالية

```
JFrame F=new JFrame("تعلم البرمجة من البداية حتى الإحتراف");
```

لاحظ في السطر السابق قمنا بإنشاء كائن من الكلاس JFrame

أما الجملة المكتوبة داخل القوسين فهي ستظهر كعنوان للواجهة

الخطوة الثانية نقوم بتحديد حجم الإطار بالطريقة التالية

```
F.setSize(width, Height);
```

ال width يعني العرض

ال Height يعني الإرتفاع

فعلى سبيل المثال لو أردنا أن يكون حجم الإطار (الحاوية) بحجم

400×400

سنكتب الأمر بالطريقة التالية

```
F.setSize(400, 400);
```

بعد إنشاء الواجهة وتحديد الحجم نقوم بتفعيل خاصة الظهور للواجهة بإستخدام الدالة setVisible بالطريقة التالية

```
F.setVisible(true);
```

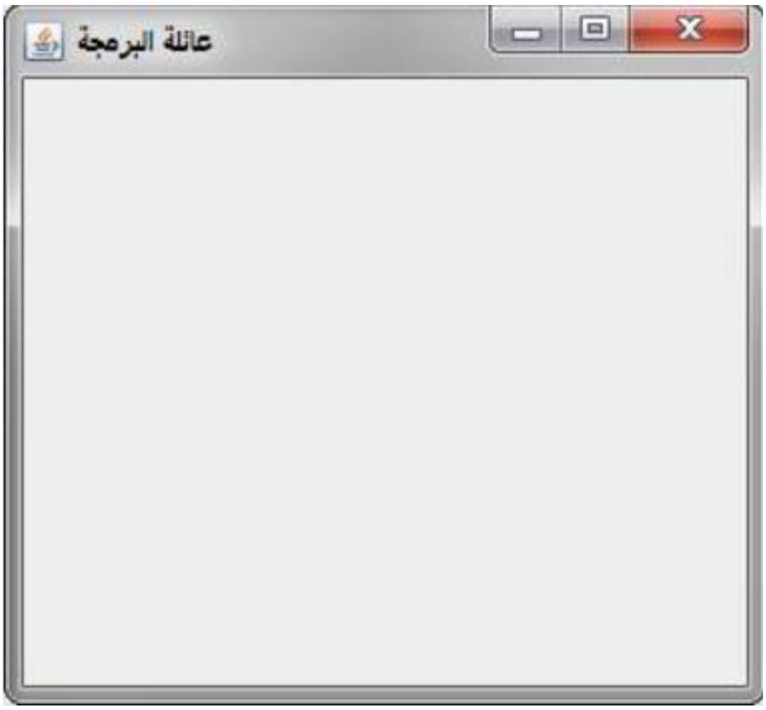
الخطوة الأخيرة قم بتنفيذ البرنامج لرؤية الواجهة

الكود كاملاً

```
package swin;

import javax.swing.*;
public class Swin {
   public static void main(String[] args) {
     JFrame F=new JFrame("تعلم البرمجة من البداية حتى الإحتراف");
     F.setSize(400,400);
     F.setVisible(true);
   }

}
```



الفصل الثالث برمجة الواجهات
الدرس السادس والعشرون كيفية إنشاء الأزرار ودمجها مع الواجهة
1 – لإنشاء زر نستخدم الكلاس JButton بالشكل التالي

```
JButton B=new JButton();
```

في السطر السابق قمنا بإنشاء كائن من الكلاس JButton حتى نستطيع الوصول إلى خصائص الكلاس JButton والتعديل عليها مثل الحجم وموقع الزر واللون والعنوان
2 – الخطوة الثانية نقوم بإستخدام الأمر setBounds الذي من خلاله نستطيع تحديد حجم الزر (العرض – الإرتفاع) و موقع الزر بالشكل التالي

```
B.setBounds(x, y, width, height);
```

حيث:
ال x تعني تحديد موقع الزر في الواجهة من اليمين إلى اليسار
ال y تعني تحديد موقع الزر في الواجهة من الأسفل إلى الأعلى
ال width تعني تحديد عرض الزر
ال height تعني تحديد ارتفاع الزر
لكتابة عنوان في الزر نستطيع فعل ذلك عند إنشاء الكائن بالشكل التالي

```
JButton B=new JButton("Click");
```

لدمج الزر في الواجهة نستخدم الأمرين التالية :

```
add(B);
setLayout(null);
```

تستخدم الدالة add التابعة للكلاس JFrame لإضافة الزر بعد أنشاءه وتحديد إحداثياته
وتستخدم الدالة SetLayout لتنفيذ وتطبيق إحداثيات الزر وحجمه لأن هذه الدالة هي التي تتحكم بالعناصر داخل الواجهة وبدونها لن يتم تنفيذ الإحداثيات والحجم
الكود التالي يوضح كيفية إنشاء زر وإظهاره داخل الواجهة JFrame
الخطوة الأولى قمنا بإنشاء واجهة بالأمر التالي

```
JFrame F=new JFrame("عائلة البرمجة");
```

بعد ذلك قمنا بإنشاء زر Button بالأمر التالي

```
JButton B=new JButton("click");
```

بعد ذلك تم تحديد الإحداثيات بواسطة الدالة setBounds باالشكل التالي

```
B.setBounds(250,100,100, 40);
```

أخيراً تم دمج الزر مع الواجهة Frame بالشكل التالي

```
F.add(B);
```

بعد ذلك تم استخدام الأمر setLayout للتحكم وتنفيذ الأحجام والأبعاد التي وضعنها في الزر بالشكل التالي

```
F.setLayout(null);
package swin;

import javax.swing.*;
public class Swin {
  Swin()
   {

    JFrame F=new JFrame("عائلة البرمجة");
```

```
 JButton B=new JButton("click");
 B.setBounds(250,100,100, 40);

  F.add(B);
  F.setLayout(null);
  F.setSize(400,400);
  F.setVisible(true);
  F.setResizable(false);

}
public static void main(String[] args) {

  Swin ob= new Swin();

}
```

- 1

الفصل الثالث : برمجة الواجهات

الدرس السابع والعشرون إضافة الحدث ActionListener

إضافة الحدث

ActionListener

المقصود بالحدث هو تنفيذ حدث معين بواسطة مؤشر الموس أو بواسطة لوحة المفاتيح
قد يتم تنفيذ الحدث عند النقر على الماوس أو عند النقر على زر معين من لوحة المفاتيح مثل الضغط على الزر Enter او بمجرد تمرير الماوس وتتنوع الأحداث حسب الرغبة
سنتعلم اليوم كيفية صنع حدث معين عند الضغط على الزر في لغة جافا
ولتنفيذ الحدث ينبغي علينا أولاً إستخدام المكتبة التالية التابعة للحزمة awt

```
import java.awt.event.*;
```

ولإرسال الحدث وتنفيذه نستخدم الحدث التالي

```
addActionListener
```

في هذا الدرس سوف يكون لدينا التالي

واجهة Frame تحتوي على زر Button وتحتوي على مربع نص Jtext

المطلوب هو أنه بمجرد الضغط على الزر تظهر لنا جملة في مربع النص JText

الخطوة الأولى سوف نقوم بتضمين المكتبتين التاية

```
import java.awt.event.*;
import javax.swing.*;
```

بعد ذلك نقوم بإنشاء الزر وتحديد الحجم والإحداثيات كما تعلمنا في الدرس السابق

```
JButton B=new JButton("click");
B.setBounds(150,100,100, 40);
```

وبنفس الطريقة سنقوم بإنشاء مربع النص JText وتحديد إحداثياته بالطريقة التالية

```
JTextField t=new JTextField();
t.setBounds(150,50, 150,20);
```

بعد ذلك نقوم بإضافة الزر ومربع النص إلى الواجهة بالطريقة التالية

```
F.add(B);
F.add(t);
```

بعد ذلك نقوم بإضافة الحدث وعلينا قبل إضافة الحدث أن نسأل انفسنا ماهو الحدث الذي نريده

الحدث الذي نريده هو أن نضغط على الزر تظهر رسالة في مربع النص

إذاً الحدث الذي سوف نستخدمه هو حدث الضغط

أين يتم الضغط

يتم الضغط على الزر

إذاً سوف يكون الحدث بالزر طالما الضغطة سوف تكون عليه

لذلك سنستخدم الحدث الخاص بالزر وننشيء منه دالة الحدث بالشكل التالي

```
B.addActionListener((ActionEvent e) -> {
    t.setText("أهلا بكم في تعلم البرمجة من البداية حتى الإحتراف");
```

في الكود السابق إستخدمنا الحدث addActionListener((ActionEvent e) التابع للكائن B

تقوم الدالة addActionListener بتنفيذ حدث بواسطة الكلاس ActionEvent المكتوب بين القوسين بعد ذلك نقوم بكتابة الحدث المطلوب وهو إظهار رسالة داخل مربع نص إذاً سوف نستخدم الكائن الخاص بمربع النص بالشكل التلي

```
t.setText("أهلا بكم في تعلم البرمجة من البداية حتى الإحتراف");
```

أي أنه عندما يتم الضغط على الزر أظهر الجملة المكتوبة بين القوسين داخل مربع النص الخاص بالكائن t

الكود كاملاً

```
package swin;
import java.awt.event.*;
import javax.swing.*;
public class Swin {
  Swin()
   {
    JFrame F=new JFrame("تعلم البرمجة من البداية حتى الإحتراف");
    JButton B=new JButton("click");
    B.setBounds(150,100,100, 40);
   JTextField t=new JTextField();
   t.setBounds(100,50, 250,20);
     F.add(B);
     F.add(t);
     F.setLayout(null);
     F.setSize(400,400);
     F.setVisible(true);
     F.setResizable(false);
    B.addActionListener((ActionEvent e) -> {
       t.setText("أهلا بكم في تعلم البرمجة من البداية حتى الإحتراف");
    });

   }
   public static void main(String[] args) {

     Swin ob= new Swin();

   }

}
```

- لإضافة لافتة أو Label إلى الواجهة نستخدم الكلاس JLabel بعد ذلك نشتق منه كائن Object كما هو الحال مع الزر

ولإنشاء لافتة JLabel نستخدم كالتالي

```
JLabel L=new JLabel ();
```

بعد ذلك نحدد الإحداثيات كالتالي

```
L.setBounds(x, y, width, hieght);
```

حيث ال

- x تعني إحداثيات أو موقع العنصر من اليمين إلى اليسار أو مايسمى بالمحور السيني
- y تعني إحداثيات أو موقع العنصر من الأسفل إلى الأعلى أو مايسمى بالمحور الصادي
- width تعني تحديد حجم العرض
- height تعني تحديد حجم الإرتفاع

في المثال المرفق مع الشرح نقوم بإنشاء واجهة وإضافة مربع نصوص JTextField و لافتة JLabel

فكرة الموضع أن المستخدم يضع عنوان أي موقع بصيغة www.webname.com

بعد ذلك يضغط المستخدم على الزر فيظهر ال IP الخاص بالموقع الذي أدخله المستخدم في اللافتة JLabel

محتوى الفكرة كالتالي

أولاً نقوم بتخزين الموقع الذي أدخله المستخدم داخل متغير نصي كالتالي

```
String host=tf.getText();
```

الخطوة الثانية نقوم بإيجاد ال IP الخاص بالموقع عن طريق الكلاس

```
InetAddress.getByName(host).getHostAddress();
```

حيث نقوم بوضع المتغير host الذي أدخله المستخدم داخل دالة البناء getByName

الكود كاملاً

```
package swin;
import java.awt.event.*;
import java.net.UnknownHostException;
import javax.swing.*;
public class Swin {
   private int x1,x2,x3;
  Swin()
   {
    JFrame F=new JFrame("تعلم البرمجة من البداية");
    JButton B=new JButton("IP أظهر  ");
     JTextField tf=new JTextField();
    JLabel L=new JLabel();
    B.setBounds(150,150,100, 40);
    tf.setBounds(50,50, 150,20);
    L.setBounds(100, 60, 300, 30);

     F.add(B);
     F.add(L);
     F.add(tf);
```

```
      F.setLayout(null);
      F.setSize(400,400);
      F.setVisible(true);
      F.setResizable(false);
      B.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
   try{
       String host=tf.getText();
       String
ip=java.net.InetAddress.getByName(host).getHostAddress();
       L.setText(  "الأيبي الخاص بالموقع"+host+" الذي أدخلته هو " + ip );
     //  L.setText("IP of "+host+" is: "+ip);
       }catch(UnknownHostException ex){System.out.println(ex);}
   }



   });
   }
   public static void main(String[] args) {

     Swin ob= new Swin();

   }

}
```



الفصل الثالث : برمجة الواجهات بإستخدام Swing
الدرس الثلاثون تطبيق عملي تعدد الأحداث
تطبيق عملي العمليات الحسابية – و +
تطبيق عملي العمليات الحسابية – و +
في هذا الدرس سوف نتعلم كيفية إنشاء أكثر من حدث في الواجهة الواحدة
أيضاً : سنتعلم شيئاً جديداً وهو اختصار كود دالة الحدث
في الدرس السابق كتبناها بهذا الشكل

```
B.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
try{
String host=tf.getText();
String ip=java.net.InetAddress.getByName(host).getHostAddress();
L.setText( "الأيبي الخاص بالموقع"+host+" الذي أدخلته هو " + ip );
// L.setText("IP of "+host+" is: "+ip);
}catch(UnknownHostException ex){System.out.println(ex);}
}
});
```

اليوم سنتعلم كيفية اختصار كود الحدث بطريقة عن طريق جعل الكلاس يرث من ال interface الخاص بالأحداث واسمه ActionListener بالطريقة التالية
لنفترض أن لدينا كلاس اسمه calc ونريد أن نجعله يرث من ActionListenerr طبعاً ال ActionListenerهو interface المخزن بداخله الأحداث وكما تعلمنا في دروس الكائنات الموجهة الفصل الثاني بأننا عندما نريد أن نجعل كلاس يرث من interface يكون بالطريقة التالية
Class Class_Name Implemnts interfaceName
في هذا الدرس لدينا واجهة تحتوي على عدد اثنين ازرار 2 Commandssكما تشاهد في الصورة احدهما للجمع والآخر للطرح
ولدينا مربعات نصوص عدد 3او مايسمى Textfield
في المربعين الأولين نضع العددين وفي المربع الثالث يتم وضع الناتج
كما تشاهد في الصورة
المطلوب هو ادخال عددين
في حال ضغطنا على الزر جمع يقوم بجمع العدد ووضع الناتج في Label ونفس الطريقة في الطرح
نحن تعلمنا كيفية إنشاء الواجهة ومربعات النص والأزرار ولا نريد أن نكررها في كل درس مايهمنا في هذا الدرس هو دالة الحدث

```
public void actionPerformed(ActionEvent ee) {
String s1=tf1.getText();
String s2=tf2.getText();
int a=Integer.parseInt(s1);
int b=Integer.parseInt(s2);
int c=0;
if(e.getSource()==b1){
c=a+b;
}else if(e.getSource()==b2){
c=a-b;
}
String result=String.valueOf(c);
```

```
tf3.setText(result);
}
```

دالة actionPerformed(ActionEvent e) تحتوي كلاس الحدث ActionEventt من هذا الكلاس يتم اشتقاق كائه اسمه e
نستفيد من الكائن e اننا بواسطته نستطيع تمييز الأحداث ونستطيع معرفة الزر التي تم الضغط عليه
ومن المهم أيضاً ان نعرف اننا بمجرد الضغط على زر فإنه سينفذ السطر التالي فمثلاً إذا ضغطنا على الزر الخاص بالكائن b1 فإنه سينفذ السطر التالي

```
b1.addActionListener(thiss);
```

الذي بدوره سوف يستدعي الدالة actionPerformed مرسلاً إليها البارامتر thiss الذي بواسطته نستطيع ان نميز ماهو الزر الذي تم الضغط عليه
وكما تلاحظ في الدالة
فإنه يتم تمييز الزر التي تم الضغط عليه بالطريقة التالية

```
if(e.getSource()==b11){
c=a+b;
}else if(e.getSource()==b2){
c=a-b;
}
```

بواسطة الدالة getSource الخاصة بالكائن e المشتق من الكلاس eventt نستطيع فحص الزر الذي تم الضغط عليه
يعني عند الضغط يقوم الزر بإستدعاء الدالة مرسلاً إليها الكائن thiss الذي يشير إلى الزر وبدروها الدالة getSource تتعرف على ماهو الزر المضغوط
الكود كاملاً في الصفحة التالية

```
package calcc;
import javax.swing.*;
import java.awt.eventt.*;
public class Calc implements ActionListener {
JTextField tf1,tf2,tf3;
JButton b1,b2;
Calc(){
JFrame f= new JFrame();
tf1=new JTextField();
tf1.setBounds(50,50,150,20);
tf2=new JTextField();
tf2.setBounds(50,100,150,20);
tf3=new JTextField();
tf3.setBounds(50,150,150,20);
tf3.setEditable(false);
b1=new JButton("+");
b1.setBounds(50,200,50,50);
b2=new JButton("-");
b2.setBounds(120,200,50,50);
b1.addActionListener(this);
b2.addActionListener(this);
f.add(tf1);f.add(tf2);f.add(tf3);f.add(b1);f.add(b2);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
public void actionPerformed(ActionEvent e) {
String s1=tf1.getText();
String s2=tf2.getText();
int a=Integer.parseInt(s1);
int b=Integer.parseInt(s2);
int c=0;
if(e.getSource()==b1){
c=a+b;
}else if(e.getSource()==b2){
c=a-b;
}
String result=String.valueOf(c);
tf3.setText(result);
}
public static void main(String[] args) {
new Calc();
}
```

```
}
```


تعلم البرمجة من البداية حتى الإحتراف
100
50
50
+
-

الفصل الثالث :	برمجة الواجهات بإستخدام swing
الدرس الواحد والثلاثون	Check Box
تطبيق عملي	برنامج كاشير مبسط
في هذا الدرس سوف نتعرف على الأشياء التالية	
• كيفية إنشاء مربع اختيار بواسطة الكلاس JCheckBox	
لإنشاء مربع اختيار نستخدم الكلاس JCheckBox بالطريقة التالية كما هو مبين في الشكل المقابل - طبعاً في الشكل المقابل تم إنشاء كائن من الكلاس JCheckBox اسمه ob هناك عدة طرق إختيارية لإستخدام الكلاس JCheckBox يمكن تلخيصها بالشكل التالي	JCheckBox ob=new JCheckBox ("")

JJCheckBox()

وتستخدم هذه الطريقة لإنشاء مربع اختيار بدون عنوان

JChechBox(String s)

وتستخدم هذه الطريقة لإنشاء مربع اختيار مع عنوان

JCheckBox(String text, boolean selected)

يمكننا أيضاً أن نعطي عنوان وحالة مربع نص مفعل أو غير مفعل

JCheckBox(Action a)

يمكننا أيضاً إنشاء مربع اختيار مع حدث

في المثال التالي قمنا بإنشاء مربع إختيار بالطريقة التالية مع أعطاء عنوان C++

```
JCheckBox checkBox1 = new JCheckBox("C++");
```

بعد ذلك قمنا بتحديد الإحداثيات (الموقع + الحجم) بالطريقة التالية

```
checkBox1.setBounds(100,90, 50,50);
```

بطبيعة الحال يكون تحديد الحجم والموقع كالتالي

القيمة الأولى تعني موقع العنصر من اليسار إلى اليمين وأعطيناها القيمة 100

القيمة الثانية تعني موقع العنصر من الأعلى إلى الأسفل وأعطيناها القيمة 90

القيمة الثالثة تعني حجم ارتفاع العنصر وأعطيناها القيمة 50

القيمة الرابعة تعني حجم عرض العنصر وأعطيناها القيمة 50

أيضاً تم إنشاء مربع اختيار أخر بالطريقة التالية وتم تحديد حالة المربع بإنه مفعل عن طريق القيمة true

```
JCheckBox checkBox2 = new JCheckBox("Java", true);
```

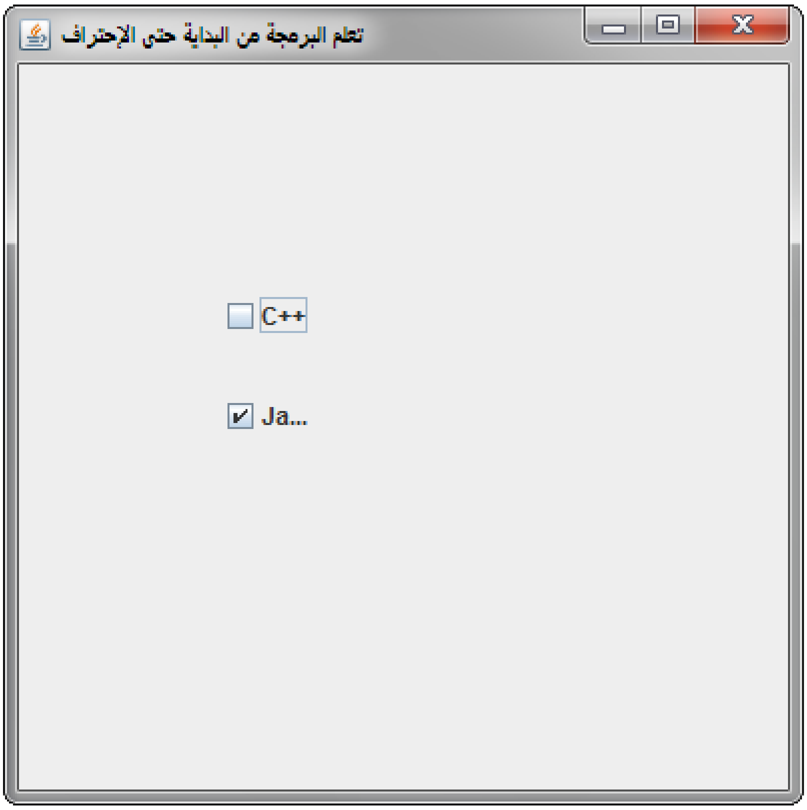
بعد ذلك تم تحديد أحداثيات وحجم العنصر كما تم شرحه في مربع النص الثاني

```
checkBox2.setBounds(100,150, 50,50);
```

الكود كاملا

```
package checkboxexample;
import javax.swing.*;
public class CheckBoxExample {
CheckBoxExample(){
      JFrame f= new JFrame("تعلم البرمجة من البداية حتى الإحتراف");
      JCheckBox checkBox1 = new JCheckBox("C++");
      checkBox1.setBounds(100,100, 50,50);
      JCheckBox checkBox2 = new JCheckBox("Java", true);
      checkBox2.setBounds(100,150, 50,50);
      f.add(checkBox1);
      f.add(checkBox2);
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
   }

   public static void main(String[] args) {
     CheckBoxExample n= new CheckBoxExample();  }}
```


تعلم البرمجة من البداية حتى الإحتراف
C++
Ja...

الدرس الثاني والثلاثون

قائمة إختيار combobox

في هذا الدرس يقوم المستخدم بإختيار قيمة من combobox ليتم طباعة ماختاره المستخدم على النموذج

- تستخدم قائمة الإختيار combobox لإظهار مجموعة من الخيارات للمستخدم

لإنشاء قائمة اختيار coboBox نستخدم الكلاس JComboBox بالشكل التالي

```
JComboBox cb=new JComboBox(Array);
```

والمقصود بـ Array هي المصفوفة التي نقوم بتعبئتها بالقيم التي تظهر في ComboBox لذا يجب علينا قبل تعريف ال comboBox أن نقوم بتعريف مصفوفة Array على سبيل المثال لنفترض أننا نريد أن نقوم بتعبئة خيارات خاصة بالمشروبات ونريدها أن تظهر في ComboBox لذا يجب علينا قبل ذلك أن نقوم بتعريف المشروبات داخل مصفوفة كالتالي

```
String a[]={"شاي عادي","شاي مغربي","قهوة عربي","","قهوة تركي"};
```

بعد ذلك نقوم بتعبئتها في ComboBox بالطريقة التالية

```
JComboBox cb=new JComboBox(a);
```

في المثال المرفق في الصورة نقوم بإظهار واجهة عندما يقوم المستخدم بإختيار مشروب يتم طباعته على Label أي طباعة المشروب الذي اختاره المستخدم عن طريق استخلاص القيمة بواسطة الدالة getItemAt(cb.getSelectedIndex()); حيث .getSelectedIndex() تعطينا رقم العنصر في ال coboBox ثم نقوم بإيجاد القيمة بواسطة getItemAt والقيمة الراجعة من هذه الدالة نقوم بتخزينه داخل متغير نصي من نوع String ثم نقوم بإظهاره على Label بحيث يصبح كالشكل التالي

```
String data = " المشروب الذي اخترته هو: "
    + cb.getItemAt(cb.getSelectedIndex());
```

3. ويتم وضع هذا الكود داخل دالة الحدث الخاص با لComboBox بحيث يصبح الحدث بالشكل التالي

```
b.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
String data = "Programming language Selected: "
    + cb.getItemAt(cb.getSelectedIndex());
label.setText(data);
```

الكود كاملاً

```
import javax.swing.*;
import java.awt.event.*;
public class ComboBox {
JFrame f;
ComboBox(){
   f=new JFrame("ComboBox Example");
   final JLabel label = new JLabel();
   label.setHorizontalAlignment(JLabel.CENTER);
   label.setSize(400,100);
   String languages[]={"شاي","شاي نعناع","قهوة عربي","قهوة تركي"};
   final JComboBox cb=new JComboBox(languages);
   cb.setBounds(50, 100,90,20);
```

```
   f.add(cb); f.add(label);
   f.setLayout(null);
   f.setSize(350,350);
   f.setVisible(true);
   cb.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
String data = "المشروب الذي اخترته هو : "
  + cb.getItemAt(cb.getSelectedIndex());
label.setText(data);
}
});
}
public static void main(String[] args) {
   new ComboBox();
}
}
```



الدرس الثالث والثلاثون

قائمة إختيار ListBox

في الدرس السابق تعرفنا على combobox مع مثال بسيط
في هذا الدرس سوف نتعرف إن شاءالله على LIstBox أو مايسمى بالقائمة وهي شبيهة جداً ب combobox
والفرق بينهما هو أن الخيارات تكون ظاهرة أمام المستخدم في LisTBox بينما تكون مختفية في combobox
ولا نستطيع مشاهدة الخيارات داخل ال combobox إلا بعد النقر عليه أما في ListBox فإنها تكون ظاهرة فقط عليه الإختيار
في هذا الدرس يقوم المستخدم بإختيار قيمة من ListBox ليتم طباعة ماختاره المستخدم على النموذج

- تستخدم قائمة الإختيار ListBox لإظهار مجموعة من الخيارات للمستخدم

ولإنشاء قائمة خيارات ListBox نمر بخطوتين الخطوة الأولى هو تعريف الخيارات ثم إنشاء ال ListBox ومن ثم دمج الخيارات داخل ListBox
بالشكل التالي
أولاً تعريف العناصر

```
DefaultListModel<String> l1 = new DefaultListModel<>();
```

بعد إنشاء كائن من الكلاس DefaultListModel نقوم بتعريف العناصر بالطريقة التالية

```
        l1.addElement("Item1");
        l1.addElement("Item2");
        l1.addElement("Item3");
        l1.addElement("Item4");
```

الخطوة الثانية نقوم بتعريف القائمة ListBox ودمج الخيارات داخلة كما ترى في السطر التالي

```
JList<String> list = new JList<>(l1);
```

.تم تعريف القائمة ListBox وتعريف كائن اسمه List وبعد ذلك تم دمج الخيارات فيه
.لأن الخيارات معرفة عن طريق الكائن l1
.بعد تعريف القائمة نقوم بتحديد الإحداثيات والحجم لهذا العنصر كما هو متعارف عليه بالطريقة التالية

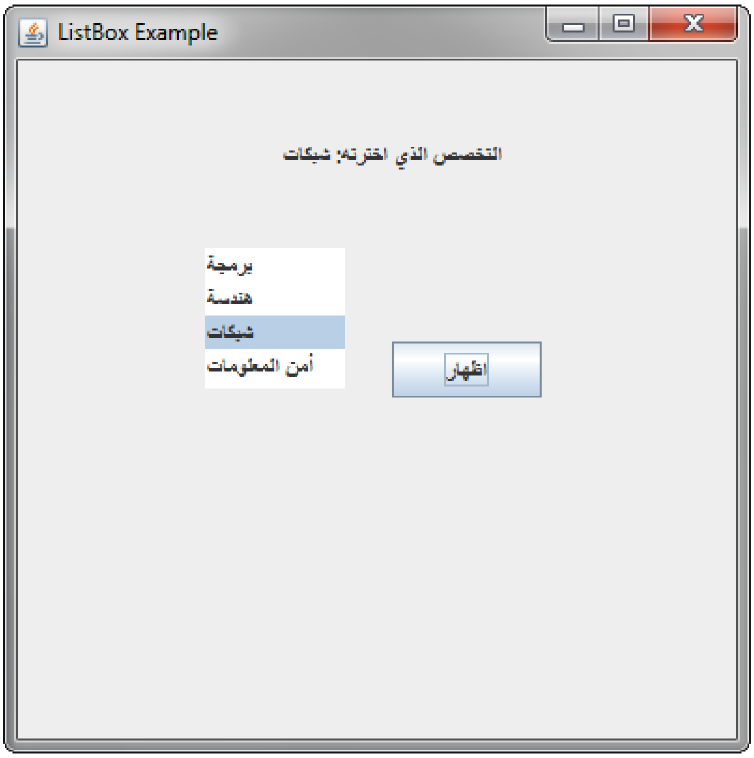
```
. list.setBounds(100,100, 75,75);
```

.في المثال التالي يقوم المستخدم بإختيار التخصص المطلوب ليظهر ماتم اختياره على الشاشة

```
import javax.swing.*;
import java.awt.event.*;
public class ListBox {
JFrame f;
ListBox(){
   f=new JFrame("ListBox Example");
   final JLabel label = new JLabel();
   label.setHorizontalAlignment(JLabel.CENTER);
   label.setSize(400,100);
      JButton b=new JButton("اظهار");
      b.setBounds(200,150,80,30);
    DefaultListModel<String> l1 = new DefaultListModel<>();
        l1.addElement("برمجة");
        l1.addElement("هندسة");
        l1.addElement("شبكات");
        l1.addElement("أمن المعلومات");
  JList<String> list = new JList<>(l1);
        list.setBounds(100,100, 75,75);
```

```
        f.add(list);
        f.add(b);
        f.add(label);

        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
         b.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
              String data = "";
              if (list.getSelectedIndex() != -1) {
                 data = "التخصص الذي اخترته: " + list.getSelectedValue();
                 label.setText(data);
              }
     label.setText(data);
           }
         });
}
public static void main(String[] args) {
    new ListBox();
}
}
```



الدرس الرابع والثلاثون

القوائم المنسدلة **JMenu**

تعتبر القوائم المنسدلة JMenu

ذات أهمية بالغة في المشاريع عن طريقها يمكننا ترتيب المحتويات وتصنيفها وتسهيل الوصول إلى كافة نماذج المشروع عن طريق تصنيفها عن طريق Menue

ولإنشاء Menu على نموذج الجافا نستخدم الكلاس نقوم بإستخدام الكلاس JMenu

نستطيع عن طريق الكلاس JMenu تعريف قائمة تظهر في أعلى النموذج

وقبل تعريف JMenu لا بد أولاً من تعريف شريط القوائم ثم دمج القائمة داخل شريط القوائم

ولتعريف شريط القائمة نستخدم الكلاس JMenuBar بالطريقة التالية

```
JMenuBar mb=new JMenuBar();
```

بعد تعريف شريط القائمة JMenuBar نستطيع أن نقوم بتعريف القائمة بالطريقة التالية

```
JMenu menu =new JMenu("Text");
```

والمقصود ب Text الذي بين القوسين هو اسم القائمة

على سبيل المثال إذا أردنا تعريف قائمة اسمها ملف نستبدل Text بكلمة ملف بالطريقة التالية

```
JMenu menu =new JMenu("ملف");
```

بعد تعريف القائمة نقوم بدمجها داخل شريط القوائم بالطريقة التالية

```
mb.add(menu);
```

هناك أمر مهم ربما تغافلنا عنه وهو العناصر التي ستظهر داخل القائمة

نستطيع أن ننشيء عناصر بواسطة الكلاس JMenuItem ثم نقوم بإنشاء كائنات بعدد العناصر التي ستظهر عند الضغط على قائمة

على سبيل المثال إذا أردنا إنشاء قائمة اسمها ملف تحتوي على العناصر التالية

فتح – حفظ – إغلاق

طالما أن لدينا ثلاثة عناصر عند ذلك سنقوم بإنشاء ثلاث كائنات بالطريقة التالية

```
JMenuItem I1,I2,I3;
```

بعد ذلك نقوم بتعريف أسماء العناصر بالطريقة التالية

```
I1=new JMenuItem("ملف");
I2=new JMenuItem(" حفظ");
I2=new JMenuItem(" إغلاق");
```

لاحظ الآن بعد تعريف العناصر سنقوم بدمجها داخل القائمة بالطريقة التالية

```
menu.add(i1); menu.add(i2); menu.add(i3);
```

الآن نستطيع القول اننا عند إنشاء قائمة مررنا بثلاث مراحل

المرحلة الأولى تعريف شريط القائمة بالكلاس

```
JMenuBar mb=new JMenuBar();
```

بعد ذلك قمنا بتعريف القائمة بالكلاس

```
JMenu menu =new JMenu("ملف");
```

بعد ذلك قمنا بتعريف العناصر التي ستظهر داخل القائمة

```
JMenuItem I1,I2,I3;
```

بعد ذلك نقوم بدمج القائمة العناصر داخل القائمة ثم دمج القائمة داخل شريط المعلومات وبهذا نكون قد أنشأنا قائمة متكاملة

```
 submenu.add(i4); submenu.add(i5);
        menu.add(submenu);
        mb.add(menu);
```

البرنامج التالي يوضح كيفية إنشاء قائمة ذات عناصر

```
package menu;
```

```
import javax.swing.*;
public class Menu {
JMenu menu;
        JMenuItem  i1, i2, i3, i4, i5;
        Menu(){
        JFrame f= new JFrame("تعلم البرمجة من البداية حتى الإحتراف");
        JMenuBar mb=new JMenuBar();
        menu=new JMenu("ملف");

        i1=new JMenuItem("فتح");
        i2=new JMenuItem("حفظ");
        i3=new JMenuItem("إغلاق");
        menu.add(i1); menu.add(i2); menu.add(i3);
        mb.add(menu);
        f.setJMenuBar(mb);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
}

   public static void main(String[] args) {
     new Menu();
   }

}
```